МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ
УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

В.О. Гороховатський, І.С. Творошенко

# МЕТОДИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ТА ОБРОБЛЕННЯ ДАНИХ

РЕКОМЕНДОВАНО
Вченою радою університету.
Протокол № 2/9 від 26.02.2021

Харків 2021

УДК 004.896

Гороховатський В.О., Творошенко І.С. Методи інтелектуального аналізу та оброблення даних: навч. посібник. – Харків: ХНУРЕ, 2021. – 92 с.

**ISBN 978-966-659-298-2**
**DOI: 10.30837/978-966-659-298-2**

У навчальному посібнику у вигляді лабораторного практикуму викладено методи вирішення ряду інтелектуальних задач, що постають у аналізі даних та при створенні систем штучного інтелекту. Для реалізації прикладних задач Data Mining запропоновано використовувати програмні пакети Statistica, MatLab та мову програмування Python, для задач штучного інтелекту – сучасне програмне середовище Visual Prolog.

Рекомендовано для здобувачів першого, другого та третього рівнів вищої освіти усіх форм навчання за спеціальністю «Комп'ютерні науки», а також для дослідників, які працюють у галузі «Інформаційні технології».

**Рецензенти:**

***В.О. Філатов***, д-р техн. наук, проф., завідувач кафедри штучного інтелекту Харківського національного університету радіоелектроніки;

***О.А. Винокурова***, д-р техн. наук, проф., головний науковий співробітник проблемної науково-дослідної лабораторії Харківського національного університету радіоелектроніки.

**ISBN 978-966-659-298-2**
**DOI: 10.30837/978-966-659-298-2**

**ЗМІСТ**

**ВСТУП**…… 5
**1 ОСНОВИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ**……..... 7
1.1 Перевірка статистичних гіпотез…….…….…….. 7
1.1.1 Теоретичні засади…….…… 8
1.1.2 Приклад перевірки статистичної гіпотези..……..... 10
1.1.3 Завдання для індивідуального виконання роботи……..…..... 15
1.1.4 Контрольні запитання та завдання…….…… 15
1.2 Кореляція і регресія…….…….……..... 16
1.2.1 Теоретичні засади…….…… 17
1.2.2 Приклад статистичного аналізу..…….…….... 18
1.2.3 Завдання для індивідуального виконання роботи……..…..... 21
1.2.4 Контрольні запитання та завдання…….…… 22
1.3 Побудова правил класифікації та дерев рішення..……. 23
1.3.1 Теоретичні засади…….…… 24
1.3.2 Приклад реалізації алгоритму ID3…….…… 30
1.3.3 Завдання для індивідуального виконання роботи……..…..... 36
1.3.4 Контрольні запитання та завдання…….…… 37
1.4 Формування асоціативних правил..…….. 38
1.4.1 Теоретичні засади…….…… 38
1.4.2 Приклад реалізації асоціативного аналізу..……..…….. 39
1.4.3 Завдання для індивідуального виконання роботи……..…..... 40
1.4.4 Контрольні запитання та завдання…….…… 41
1.5 Застосування кластерного аналізу…….…… 42
1.5.1 Теоретичні засади…….…… 42
1.5.2 Приклад реалізації методу кластеризації...……..…….. 44
1.5.3 Завдання для індивідуального виконання роботи……..…..... 50
1.5.4 Контрольні запитання та завдання…….…… 50
1.6 Комплекс завдань з основ інтелектуального аналізу даних.…….. 51
**2 ВИРІШЕННЯ ІНТЕЛЕКТУАЛЬНИХ ЗАДАЧ**……..… 52
2.1 Розпізнавання візуальних об’єктів у системах комп’ютерного зору.… 52
2.1.1 Теоретичні засади…….…… 52

2.1.2 Особливості розпізнавання візуальних об’єктів..... 55
2.1.3 Завдання для індивідуального виконання роботи..... 56
2.1.4 Контрольні запитання та завдання..... 56
2.2 Пошук рішень інтелектуальних задач..... 57
2.2.1 Теоретичні засади..... 57
2.2.2 Завдання для індивідуального виконання роботи..... 61
2.2.3 Контрольні запитання та завдання..... 61
2.3 Прийняття рішень в експертних системах..... 62
2.3.1 Теоретичні засади..... 62
2.3.2 Приклади обчислення та візуалізації значень експертних оцінок..... 64
2.3.3 Завдання для індивідуального виконання роботи..... 66
2.3.4 Контрольні запитання та завдання..... 67
2.4 Моделювання дій з нечіткими числами та множинами..... 68
2.4.1 Теоретичні засади..... 68
2.4.2 Завдання для індивідуального виконання роботи..... 73
2.4.3 Контрольні запитання та завдання..... 74
2.5 Робота з базами знань у Visual Prolog. Обчислення виразів..... 75
2.5.1 Теоретичні засади..... 75
2.5.2 Приклади вирішення конкретних задач роботи з базою знань..... 77
2.5.3 Завдання для індивідуального виконання роботи..... 81
2.5.4 Контрольні запитання та завдання..... 81
2.6 Програмування рекурсивних викликів. Робота зі списками..... 82
2.6.1 Приклади програмування рекурсивних викликів..... 82
2.6.2 Завдання для індивідуального виконання роботи..... 84
2.6.3 Контрольні запитання та завдання..... 84
2.7 Вирішення логічних задач у Visual Prolog..... 85
2.7.1 Приклади вирішення логічних задач..... 85
2.7.2 Завдання для індивідуального виконання роботи..... 88
2.7.3 Контрольні запитання та завдання..... 88
**ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ**..... 89

# ВСТУП

Сучасні методи *інтелектуального аналізу даних* (ІАД, Data Mining, Data science, Knowledge Discovery in Databases, здобування даних, виявлення знань та закономірностей у даних) охоплюють такі задачі, як перетворення, зберігання, аналіз, моделювання та отримання інформації при прийнятті рішень на основі фактичних даних [1]. Data Mining вивчає процес знаходження нових та потенційно корисних знань у базах різноманітних даних [2]. Це комплексний науковий напрям, що знаходиться на перетині таких наук: системи баз даних, статистика, штучний інтелект, дискретна математика, комп'ютерна лінгвістика, теорія графів, теорія алгоритмів тощо [3]. Математичний та статистичний підходи є основою для побудови методів Data Mining [4].

*Методи та системи штучного інтелекту* – сучасна наука, що знаходить все більше поширення та прикладне застосування у різноманітних інформаційних системах прийняття інтелектуальних рішень [5,6]. Завданням цієї науки є забезпечення розумних суджень і дій за допомогою обчислювальних систем та інших штучних пристроїв [7].

*Штучний інтелект* – в значній мірі експериментальна наука: створюючи ті чи інші комп'ютерні уявлення і програмні моделі, дослідник порівнює їх поведінку між собою та з прикладами вирішення тих же задач людиною-фахівцем, модифікує їх на основі порівняння, щоб добитися кращої відповідності результатів [8].

Навчальний посібник «Методи інтелектуального аналізу та оброблення даних» у вигляді лабораторного практикуму складається із двох розділів, які присвячені впровадженню методів інтелектуального аналізу даних та вирішенню інтелектуальних задач.

У першому розділі посібника висвітлено як традиційні методи (перевірка гіпотез, кореляція, регресія, кластеризація), так і сучасні методи та засоби (побудова класифікаційних правил, дерев рішень, асоціативних правил) інтелектуального аналізу даних. Для вирішення задач Data Mining запропоновано використання програмних пакетів Statistica та MatLab, а також мову програмування Python.

У другому розділі навчального посібника використовуються традиційні методи (програмування і пошук рішень у середовищі Visual Prolog) та популярні методи і засоби (алгоритми розпізнавання у комп'ютерному зорі, нечіткі множини) вирішення інтелектуальних задач. Для вирішення задач штучного інтелекту запропоновано сучасні програмні середовища.

*Метою* навчального посібника є сформувати у студентів комплекс теоретичних знань та практичних навичок щодо прикладного впровадження методів інтелектуального аналізу даних та методів штучного інтелекту.

*Завданням* навчального посібника є продемонструвати методологію вирішення інтелектуальних задач за допомогою використання сучасних комп'ютерних технологій і програмних засобів, що реалізують методи інтелектуального аналізу даних та штучного інтелекту.

У результаті детального опрацювання навчального посібника здобувач отримає *компетентності* пов'язані із здатністю:

– до виявлення статистичних закономірностей недетермінованих явищ, застосування методів обчислювального інтелекту, зокрема статистичної, нейромережевої та нечіткої обробки даних, методів машинного навчання та генетичного програмування тощо;

– до інтелектуального аналізу даних на основі методів обчислювального інтелекту включно з великими та погано структурованими даними, їхньої оперативної обробки та візуалізації результатів аналізу в процесі розв'язування прикладних задач;

– застосовувати існуючі і розробляти нові алгоритми розв'язування задач у галузі комп'ютерних наук: алгоритми розв'язання обчислювальних та логічних задач, алгоритми паралельних та розподілених обчислень, алгоритми аналітичної обробки й інтелектуального аналізу великих даних з оцінкою їх ефективності та складності.

Здобувачі повинні продемонструвати *знання* щодо методів і алгоритмів аналітичної обробки та інтелектуального аналізу великих масивів даних для задач класифікації, прогнозування, кластерного аналізу, пошуку асоціативних правил з використанням програмних інструментів підтримки аналізу даних та прийняття рішень.

Здобувачі повинні продемонструвати *вміння*:

– використовувати методи обчислювального інтелекту, машинного навчання, нейромережевої та нечіткої обробки даних, генетичного та еволюційного програмування для розв'язання задач розпізнавання, прогнозування, класифікації, ідентифікації об'єктів керування тощо;

– використовувати технології Data Mining, Text Mining, Web Mining для інтелектуального аналізу даних, краудсорсінгу, інтеграції різнорідних даних з різних джерел для глибинного аналізу, машинного навчання, отримання прогнозів на основі базових моделей, штучних нейронних мереж, для розпізнавання образів тощо.

# 1 ОСНОВИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

## 1.1 Перевірка статистичних гіпотез

*Статистична гіпотеза* – це будь-яке твердження, що стосується розподілу деякої випадкової величини чи події, яке перевіряється на підставі її вибіркових значень [9].

Статистичні гіпотези носять характер правдоподібного твердження, яке має цілком визначену ймовірність. Всі висновки про правильність чи помилковість статистичних гіпотез, які можна висловити на основі скінченного числа спостережень, носять імовірнісний характер. Якщо не знайдено протиріч, на основі яких гіпотеза повинна бути відхилена, то статистична гіпотеза відкрита для подальшої перевірки.

*Нульова гіпотеза* – це початкова гіпотеза, яка передбачає, що між параметрами генеральних сукупностей немає очікуваних розбіжностей, її прийнято позначати $H_0$.

*Альтернативна гіпотеза* (або конкуруюча) – це гіпотеза, яка передбачає, що варіації генеральних сукупностей статистично неоднакові, її прийнято позначати $H_A$.

Точність перевірки гіпотези вибирається з практичних міркувань.

Відповідна похибка визначається рівнем значущості $\alpha$.

При перевірці гіпотез помилки розрізняють за двома видами:

– *помилки першого роду* (рівень значущості) – відхилення істинного припущення;

– *помилки другого роду* (потужність критерію) – прийняття помилкового твердження.

Процедура перевірки гіпотез зводиться до обчислення за вибірковими даними величини, яку називають *статистичним критерієм*.

*Критерій* – це математичний метод, який дозволяє перевірити статистичну гіпотезу і забезпечує прийняття істинної і відхилення неправильної гіпотези з наперед заданою імовірністю.

*Статистичні критерії* значущості бувають трьох типів:

1. Параметричні критерії, які оперують фізичними величинами (м, кг, с).

2. Непараметричні критерії, в яких використовують величини, що не мають розмірностей фізичних величин (місця, ранги).

3. Критерії узгодженості, які використовують для перевірки узгодженості розподілу генеральної сукупності з прийнятою раніше теоретичною моделлю (наприклад, з нормальним розподілом).

Найчастіше на практиці використовують, як критерії, нормальний розподіл, $\chi^2$-розподіл, розподіли Стьюдента або Фішера. Значенням критерію, що спостерігається, називають його величину, яку розраховують за досліджуваними вибірками.

Для перевірки гіпотези весь вибірковий простір поділяють на дві області, що не перетинаються: область прийняття та критичну.

*Областю прийняття гіпотези* (областю допустимих значень) називають сукупність значень критерію, за яких нульову гіпотезу приймають. Перевірка гіпотези передбачає розрахунок значення критерію і перевірку його потрапляння до області прийняття гіпотези.

*Критичною областю* називають сукупність значень критерію, за яких нульову гіпотезу слід відхилити.

### 1.1.1 Теоретичні засади

Основні поняття теорії перевірки статистичних гіпотез, що використані при виконанні даного завдання: статистична гіпотеза, нульова та альтернативна гіпотези, помилки першого та другого роду, статистичний критерій, область прийняття гіпотези, критична область [9].

Розглянемо застосування критерію $\chi^2$ для перевірки гіпотези про закон розподілу. Нехай $x_1, x_2, ..., x_n$ – вибірка спостережень випадкової величини *X*. Перевіряється гіпотеза $H_0$, яка стверджує, що *X* має функцію розподілу *F*(*X*).

Перевірка гіпотези $H_0$ за допомогою критерію $\chi^2$ здійснюється за наступною схемою. За вибіркою спостережень знаходять оцінки невідомих параметрів передбачуваного закону розподілу випадкової величини *X*. Далі, область значень випадкової величини *X* розбивається на *r* множин $\Delta_1, \Delta_2, \Delta_r$, наприклад, *r* інтервалів у випадку, коли *X* – безперервна випадкова величина, або *r* груп, що складаються з окремих значень, для дискретної випадкової величини *X*.

Нехай $n_k$ – число елементів вибірки, що належать множині $\Delta_k$, $k = 1, 2, ..., r$, $\sum_{k=1}^{r} n_k = n$.

Використовуючи передбачуваний закон розподілу випадкової величини $X$, знаходять імовірності $p_k$ того, що значення $X$ належить множині $\Delta_k$, тобто $p_k = P[X \in \Delta_k]$, $k = 1, 2, ..., r$, $\sum_{k=1}^{r} p_k = 1$. Результати можна подати у вигляді таблиці 1.1.

Таблиця 1.1 – Результати обчислень

| | Число спостережень | | | | Всього |
|---|---|---|---|---|---|
| | $\Delta_1$ | $\Delta_2$ | … | $\Delta_r$ | |
| Спостережуване | $n_1$ | $n_2$ | … | $n_r$ | $n$ |
| Очікуване | $np_1$ | $np_2$ | … | $np_r$ | $n$ |

Вибіркове значення статистики критерію $\chi^2$ обчислюється за формулою:

$$\chi^2_{\hat{a}} = \sum_{k=1}^{r} \frac{(n_k - np_k)^2}{np_k}. \qquad (1.1)$$

Гіпотеза $H_0$ узгоджується з результатами спостережень на рівні значущості $\alpha$, якщо

$$\chi^2_{\hat{a}} < \chi^2_{1-\alpha}(r - l - 1), \qquad (1.2)$$

де $\chi^2_{1-\alpha}(r-l-1)$ – квантиль порядку $1-\alpha$ розподілу $\chi^2$ з $(r-l-1)$ ступенями волі, а $l$ – число невідомих параметрів розподілу, що оцінюються за вибіркою; якщо ж $\chi^2_{\hat{a}} < \chi^2_{1-\alpha}(r-l-1)$, то гіпотеза $H_0$ відхиляється.

**Зауваження.** Критерій $\chi^2$ використовує той факт, що випадкові величини $\frac{n_k - np_k}{\sqrt{np_k}}$, $k = 1, 2, ..., r$, мають розподіли, близькі до нормального $N(0,1)$. Щоб це твердження було достатньо точним, необхідно, щоб для всіх інтервалів виконувалась умова $np_k \geq 5$. Якщо для деяких інтервалів ця умова не виконується, то їх слід об'єднати з сусідніми. Статистичні пакети виконують це автоматично.

### 1.1.2 Приклад перевірки статистичної гіпотези

Перевіримо гіпотезу про нормальний розподіл за вибіркою із $n = 55$ спостережень, що мають значення, наведені у таблиці 1.2. Приймемо $\alpha = 0{,}1$.

Таблиця 1.2 – Статистичні дані

| | | | | | | |
|---|---|---|---|---|---|---|
| 18,3 | 15,4 | 17,2 | 19,2 | 23,3 | 18,1 | 21,9 |
| 15,3 | 16,8 | 13,2 | 20,4 | 16,5 | 19,7 | 20,5 |
| 14,3 | 20,1 | 16,8 | 14,7 | 20,8 | 19,5 | 15,3 |
| 19,3 | 17,8 | 16,2 | 15,7 | 22,8 | 21,9 | 12,5 |
| 10,1 | 21,1 | 18,3 | 14,7 | 14,5 | 18,1 | 18,4 |
| 13,9 | 19,1 | 18,5 | 20,2 | 23,8 | 16,7 | 20,4 |
| 19,5 | 17,2 | 19,6 | 17,8 | 21,3 | 17,5 | 19,4 |
| 17,8 | 13,5 | 17,8 | 11,8 | 18,6 | 19,1 | |

**Рішення 1.** Для перевірки гіпотези про нормальний розподіл знайдемо оцінки математичного очікування і дисперсії:

$$\tilde{m} = \tilde{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \approx 17{,}87\,, \tag{1.3}$$

$$\tilde{\sigma}^2 = s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \approx 8{,}62\,. \tag{1.4}$$

Згрупуємо вибірку, розширивши перший та останній інтервали. Результати групування вибірки наведені в другому та третьому стовпчиках таблиці 1.3.

У четвертому стовпчику таблиці 1.3 наведені ймовірності $p_k$, що обчислюються за формулою

$$p_k = P[X \in \Delta_k] = \Phi\left(\frac{b_k - \bar{x}}{s}\right) - \Phi\left(\frac{a_k - \bar{x}}{s}\right),\ k = 1, 2, \ldots, 7\,, \tag{1.5}$$

де $a_k$ і $b_k$ – відповідно нижня та верхня межі інтервалу $\Delta_k$.

Значення функції Лапласа $\Phi(x)$ беруться з відповідної таблиці (або з використанням ймовірнісного калькулятора).

Таблиця 1.3 – Результати групування вибірки

| № інтервалу | Межі інтервалу | Спостережувана частота | Імовірність потрапляння в інтервал | Очікувана частота | Обробка отриманих даних | | |
|---|---|---|---|---|---|---|---|
| $k$ | $\Delta_k$ | $n_k$ | $\Delta_k,\ p_k$ | $np_k$ | $np_k$ | $n_k - np_k$ | $\frac{(n_k - np_k)^2}{np_k}$ |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| 1 | –∞–12 | 2 | 0,0228 | 1,254 | 5,274 | 0,725 | 0,010 |
| 2 | 12–14 | 4 | 0,0731 | 4,020 | | | |
| 3 | 14–16 | 8 | 0,1686 | 9,273 | 9,273 | –1,273 | 0,175 |
| 4 | 16–18 | 12 | 0,2576 | 14,168 | 14,168 | –2,168 | 0,332 |
| 5 | 18–20 | 16 | 0,2484 | 13,662 | 13,662 | –2,338 | 0,400 |
| 6 | 20–22 | 10 | 0,1519 | 8,354 | 12,633 | 0,366 | 0,011 |
| 7 | 22–+∞ | 3 | 0,0778 | 4,279 | | | |
| | Сума | 55 | 1,0001 | 55 | 55 | – | 0,928 |

В п’ятому стовпчику таблиці 1.3 наведені очікувані частоти $np_k$, а в шостому – значення $np_k$ після об’єднання перших двох та останніх двох інтервалів.

Оскільки після об’єднання залишилось $r = 5$ інтервалів, а за вибіркою визначені оцінки двох параметрів – математичного очікування і дисперсії, тобто $l = 2$, то число вільних ступенів дорівнює $5 - 2 - 1 = 2$. За таблицею квантилів розподілу $\chi^2$ знаходимо $\chi^2_{0,90}(2) = 4{,}61$. Вибіркове значення статистики критерію дорівнює $\chi^2_a = 0{,}928$. Отже, гіпотеза про нормальний розподіл приймається.

**Рішення 2.** Розглянемо рішення даного прикладу в пакеті Statistica. Введіть вибірку в змінну Var1 (рис. 1.1).

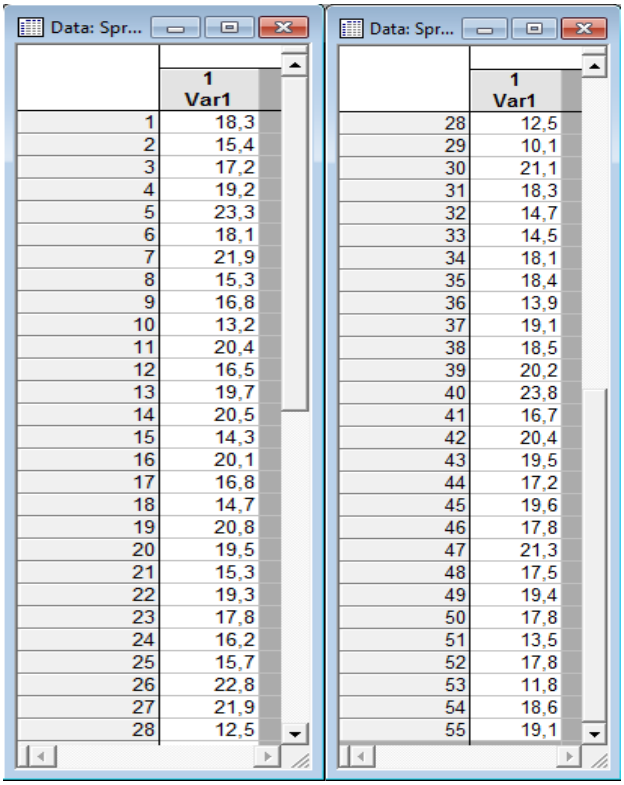



Рисунок 1.1 – Результат введення вибірки в пакет Statistica

Для перевірки гіпотези про нормальний розподіл за критерієм $\chi^2$ в пакеті Statistica виконайте наступні дії.

У меню Statistics виберіть розділ Distribution Fitting (вибір розподілу, рисунок 1.2).

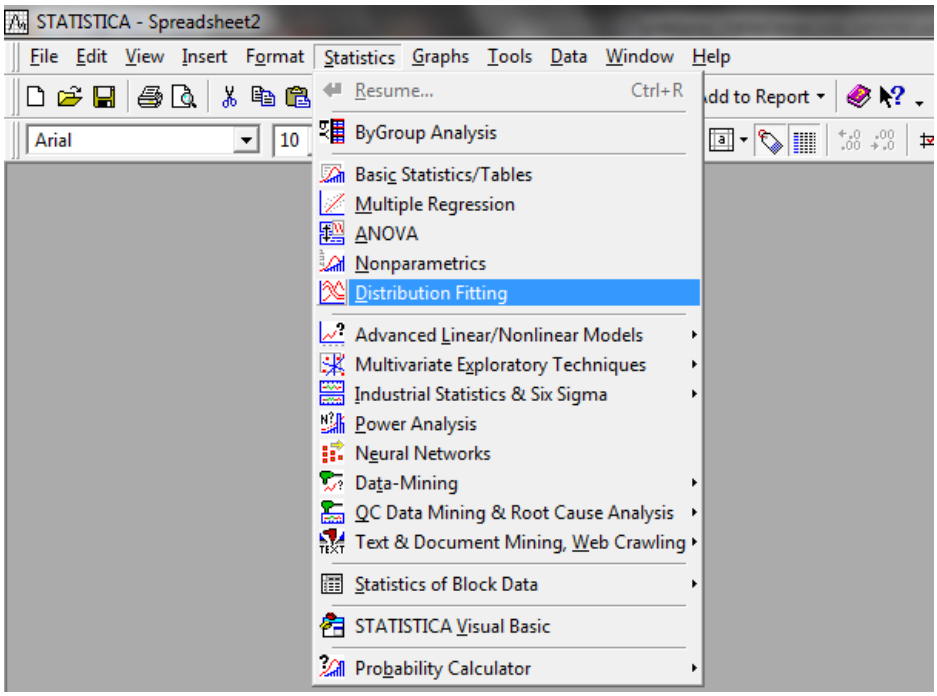



Рисунок 1.2 – Розділ Distribution Fitting в пакеті Statistica

Далі в Continuous Distributions (безперервні розподіли) виберіть Normal (нормальний розподіл, рисунок 1.3). Натисніть OK.

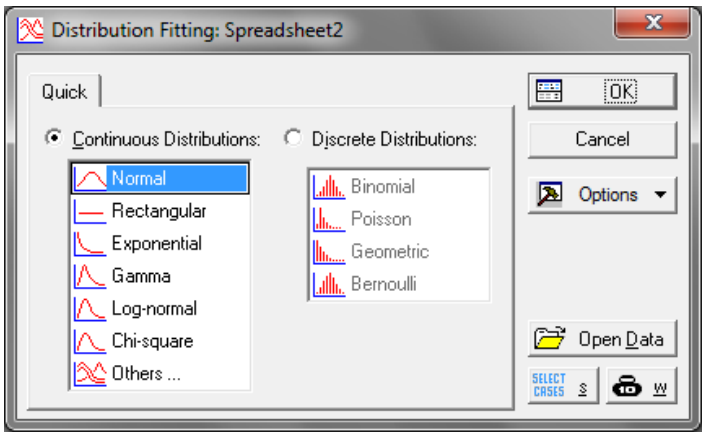



Рисунок 1.3 – Опція Continuous Distributions у розділі Distribution Fitting

В меню Fitting Continuous Distributions натисніть на кнопку Variable і виділіть змінну VAR1. Після виділення змінної на екрані з'являться оцінки математичного очікування та дисперсії. За замовчуванням число інтервалів групування дорівнює 18. Це число можна змінити відповідно зо задачі та натиснути OK.

На екран виводиться таблиця (рис. 1.4) для розрахунку статистики критерію (тест $\chi^2$ – Chi-Square test, дорівнює 1,85169), обчислене число ступенів свободи для об'єднаних інтервалів (df = 4 (adjusted)) та обчислений системою рівень значущості $p = 0{,}76301$.

| Upper Boundary | Observed Frequency | Cumulative Observed | Percent Observed | Cumul. % Observed | Expected Frequency | Cumulative Expected | Percent Expected | Cumul. % Expected | Observed-Expected |
|---|---|---|---|---|---|---|---|---|---|
| Variable: Var1, Distribution: Normal (Spreadsheet1) Chi-Square = 1,85169, df = 4 (adjusted) , p = 0,76301 | | | | | | | | | |
| <= 9,00000 | 0 | 0 | 0,00000 | 0,0000 | 0,069950 | 0,06995 | 0,12718 | 0,1272 | -0,06995 |
| 10,00000 | 0 | 0 | 0,00000 | 0,0000 | 0,134096 | 0,20405 | 0,24381 | 0,3710 | -0,13410 |
| 11,00000 | 1 | 1 | 1,81818 | 1,8182 | 0,330997 | 0,53504 | 0,60181 | 0,9728 | 0,66900 |
| 12,00000 | 1 | 2 | 1,81818 | 3,6364 | 0,728256 | 1,26330 | 1,32410 | 2,2969 | 0,27174 |
| 13,00000 | 1 | 3 | 1,81818 | 5,4545 | 1,428243 | 2,69154 | 2,59681 | 4,8937 | -0,42824 |
| 14,00000 | 3 | 6 | 5,45455 | 10,9091 | 2,496785 | 5,18833 | 4,53961 | 9,4333 | 0,50321 |
| 15,00000 | 4 | 10 | 7,27273 | 18,1818 | 3,890673 | 9,07900 | 7,07395 | 16,5073 | 0,10933 |
| 16,00000 | 4 | 14 | 7,27273 | 25,4545 | 5,404245 | 14,48324 | 9,82590 | 26,3332 | -1,40424 |
| 17,00000 | 5 | 19 | 9,09091 | 34,5455 | 6,691352 | 21,17460 | 12,16609 | 38,4993 | -1,69135 |
| 18,00000 | 7 | 26 | 12,72727 | 47,2727 | 7,385206 | 28,55980 | 13,42765 | 51,9269 | -0,38521 |
| 19,00000 | 7 | 33 | 12,72727 | 60,0000 | 7,265770 | 35,82557 | 13,21049 | 65,1374 | -0,26577 |
| 20,00000 | 9 | 42 | 16,36364 | 76,3636 | 6,371931 | 42,19750 | 11,58533 | 76,7227 | 2,62807 |
| 21,00000 | 6 | 48 | 10,90909 | 87,2727 | 4,981154 | 47,17866 | 9,05664 | 85,7794 | 1,01885 |
| 22,00000 | 4 | 52 | 7,27273 | 94,5455 | 3,471019 | 50,64968 | 6,31094 | 92,0903 | 0,52898 |
| 23,00000 | 1 | 53 | 1,81818 | 96,3636 | 2,156007 | 52,80568 | 3,92001 | 96,0103 | -1,15601 |
| 24,00000 | 2 | 55 | 3,63636 | 100,0000 | 1,193732 | 53,99942 | 2,17042 | 98,1808 | 0,80627 |
| 25,00000 | 0 | 55 | 0,00000 | 100,0000 | 0,589147 | 54,58856 | 1,07118 | 99,2519 | -0,58915 |

Рисунок 1.4 – Розрахунок статистики

Оскільки обчислений рівень $p = 0{,}76301$ перевищує значення заданого рівня значущості $\alpha = 0{,}1$, то гіпотеза про нормальний розподіл не може бути відхилена і, отже, дані не суперечать нормальному закону розподілу.

В отриманій таблиці (рис. 1.4) у стовпці:

– Observed Frequency – спостережувані частоти;

– Cumulative Observed – спостережувані накопичені частоти;

– Percent Observed – спостережувані відсотки;

– Cumul. % Observed – спостережувані накопичені відсотки;

– Expected Frequency – очікувані частоти;

– Cumulative Expected – очікувані накопичені частоти;

– Percent Expected – очікувані відсотки;

– Cumul. % Expected – очікувані накопичені відсотки;

– Observed-Expected – різниця між очікуваними частотами і тими, що спостерігаються.

На рисунку 1.5 з метою візуального аналізу наведена гістограма отриманих частот та крива нормального розподілу.

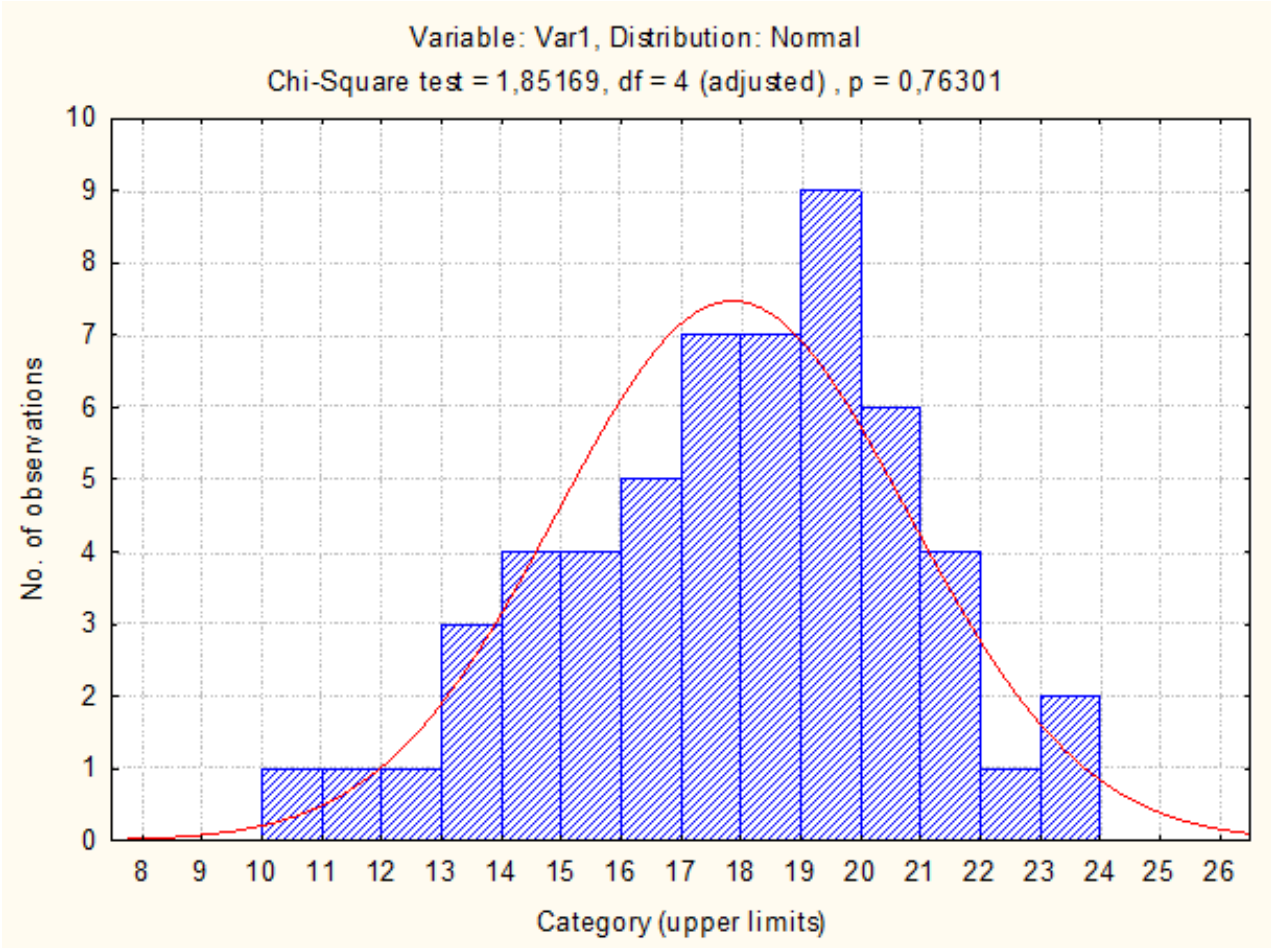



Рисунок 1.5 – Гістограма отриманих частот та крива нормального розподілу

### 1.1.3 Завдання для індивідуального виконання роботи

Сформувати програмними засобами або за допомогою пакету Statistica відповідний числовий масив даних та виконати перевірку гіпотези про нормальний закон розподілу.

Змінити так вхідні дані, щоб гіпотеза не підтвердилася. Пояснити результат.

Застосувати можливості пакету Статистика для перевірки гіпотез:

а) порівняти дисперсії двох нормальних генеральних сукупностей;

б) порівняти середні довільно розподілених генеральних сукупностей;

в) застосувати критерій $\chi^2$ для перевірки гіпотези про закон розподілу.

### 1.1.4 Контрольні запитання та завдання

1. Пояснити поняття «нульова гіпотеза» та «альтернативна гіпотеза».
2. Пояснити поняття «помилка першого роду» (рівень значущості).
3. Пояснити поняття «помилка другого роду» (потужність критерію).
4. Пояснити, як пов’язані між собою помилки першого та другого роду.
5. Пояснити, як необхідно застосовувати статистичний критерій.
6. Пояснити поняття «область прийняття гіпотези», «критична область».

## 1.2 Кореляція і регресія

Взаємозв'язок, при якому одному значенню однієї величини може відповідати декілька значень другої, називається *статистичним*. Наприклад, одному значенню зросту людини може відповідати декілька значень її маси.

Статистичний метод, який використовується для дослідження взаємозв'язків, називається *кореляційним аналізом*.

Кореляційний аналіз полягає у визначенні ступеня та виду зв'язку між двома випадковими величинами *X* і *Y*.

Дослідження характеру взаємозв'язку починається з побудови графічного зображення для результатів вимірів у прямокутній системі координат, де кожна пара результатів буде відображатись точкою. Така графічна залежність називається *діаграмою розсіяння* або *кореляційним полем* [9].

*Кореляційне поле* відображає статистичний взаємозв'язок між результатами вимірів.

Візуальний аналіз кореляційного поля дозволяє якісно оцінити форму, спрямованість і тісноту взаємозв'язку.

Форма визначається по виду кореляційного поля: якщо через кореляційне поле можна провести пряму лінію – форма зв'язку лінійна, якщо ні – нелінійна.

Для оцінювання взаємозв'язку в кореляційному аналізі використовується спеціальний показник – коефіцієнт кореляції. Абсолютне значення коефіцієнта кореляції знаходиться в межах від 0 до 1. Коефіцієнт кореляції дає кількісну оцінку статистичного взаємозв'язку між результатами вимірів.

У практичних дослідженнях виникає необхідність апроксимувати (описати наближено) діаграму розсіяння математичним рівнянням.

У прямокутній системі координат рівняння прямої лінії записується у вигляді наближеного запису статистичної залежності, який називається *рівнянням регресії*:

$$y = a + bx, \tag{1.6}$$

де $a$ – вільний член рівняння регресії;

$b$ – коефіцієнт регресії.

Коефіцієнт регресії $b$ має розмірність, яка дорівнює відношенню розмірностей показників, що вивчаються, і такий самий знак, що і коефіцієнт кореляції.

Лінію рівняння регресії будують на кореляційному полі по двох точках.

### 1.2.1 Теоретичні засади

Дві випадкові величини можуть бути незалежними або пов'язані між собою визначеною функціональної залежністю, або залежністю особливого типу, що називається статистичною (стохастичною).

*Статистичною* називають залежність, при якій зміна однієї з випадкових величин спричиняє зміну розподілу іншої випадкової величини. Статистична залежність виявляється зокрема в тому, що при зміні однієї з величин змінюється середнє значення іншої; при цьому статистичну залежність називають кореляційною.

Умовним середнім $\bar{y}_x$ називається середнє арифметичне значень випадкової величини *Y*, що спостерігаються за умови, якщо випадкова величина *X* при цьому має значення *x.* Аналогічно визначається і умовне середнє $\bar{x}_y$.

Вибіркова оцінка $\bar{y}_x$ є функцією від змінної *x*, що позначено через $f^*(x)$ і будемо називати *вибірковою регресією Y на X*, а її графік – *вибірковою лінією регресії Y на X.* Вибіркові рівняння регресії *Y* на *X* та *X* на *Y* мають вигляд:

$$\begin{aligned} \bar{y}_x &= f^*(x), \\ \bar{x}_y &= \varphi^*(y). \end{aligned} \tag{1.7}$$

*Вибірковий коефіцієнт кореляції.* Як відомо з теорії ймовірностей, якщо величини *X* і *Y* незалежні, коефіцієнт їхньої кореляції $r = 0$, якщо $r = \pm 1$ – величини *X* і *Y* пов'язані лінійною функціональною залежністю. Тобто коефіцієнт кореляції *r* характеризує ступінь лінійного зв'язку між *X* і *Y*.

*Криволінійна кореляція.* Якщо графік регресії $\bar{y}_x = f^*(x)$ або $\bar{x}_y = \varphi^*(y)$ є відмінним від прямої лінії, кореляцію називають *криволінійною.* Зокрема, функція регресії *Y* на *X* може мати вигляд багаточлена:

$$\bar{y}_x = ax^2 + bx + c. \tag{1.8}$$

Така регресія називається параболічною кореляцією другого порядку.

Для визначення вигляду функції регресії на площині наносять точки $(x, \bar{y}_x)$ і за аналізом їх розташування приходять до висновку про можливий вигляд функції регресії.

При остаточному рішенні беруть до уваги особливості задачі, яка розв'язується. Теорія криволінійної кореляції вирішує задачу знаходження невідомих параметрів рівняння регресії. Їх шукають методом найменших квадратів. Для оцінки значень криволінійної кореляції використовують вибіркові кореляційні відношення.

Припустимо, що дані *n* спостережень (вибірки) дозволяють вважати, що має місце кореляція другого порядку. У цьому випадку вибіркове рівняння регресії *Y* на *X* має вигляд: $\bar{y}_x = Ax^2 + Bx + C,$ де *A, B, C* – невідомі параметри.

Використовуючи метод найменших квадратів, одержуємо систему лінійних рівнянь:

$$\begin{cases} (\sum n_x x^4)A + (\sum n_x x^3)B + (\sum n_x x^2)C = \sum n_x \bar{y}_x x^2, \\ (\sum n_x x^3)A + (\sum n_x x^2)B + (\sum n_x x)C = \sum n_x \bar{y}_x x, \\ (\sum n_x x^2)A + (\sum n_x x)B + n_x C = \sum n_x \bar{y}_x. \end{cases} \quad (1.9)$$

звідки отримуємо значення параметрів *A, B, C* та рівняння регресії.

### 1.2.2 Приклад статистичного аналізу

Розглянемо вибірку із 24 спостережень (рис. 1.6).

Для проведення регресійного аналізу будемо використовувати пакет Statistica.

| | 1<br>x | 2<br>y |
|---|---|---|
| 1 | 5,5 | 12,8 |
| 2 | 5,3 | 12,3 |
| 3 | 5,1 | 11,2 |
| 4 | 4,9 | 10,5 |
| 5 | 4,7 | 9,7 |
| 6 | 4,5 | 9,2 |
| 7 | 4,3 | 8,3 |
| 8 | 4,1 | 7,7 |
| 9 | 3,9 | 7,1 |
| 10 | 3,7 | 6,5 |
| 11 | 3,5 | 5,9 |
| 12 | 3,3 | 5,4 |
| 13 | 3,1 | 4,9 |
| 14 | 2,9 | 4,4 |
| 15 | 2,7 | 4,1 |
| 16 | 2,5 | 3,6 |
| 17 | 2,3 | 3,2 |
| 18 | 2,1 | 2,8 |
| 19 | 1,9 | 2,5 |
| 20 | 1,7 | 2,5 |
| 21 | 1,5 | 1,9 |
| 22 | 1,3 | 1,7 |
| 23 | 1,1 | 1,5 |
| 24 | 0,9 | 1,3 |

Рисунок 1.6 – Вибірка спостережень

Для подальшого аналізу побудуємо графік залежності $Y$ від $X$.

Для цього оберемо: Graphs -> 2D Graphs -> Scatterplots (рис. 1.7) і отримаємо графік (рис. 1.8).

З графіку (рис. 1.8) можна зробити висновок, що дана залежність за зовнішнім виглядом може бути описана параболічною кривою $y = ax^2 + bx + c$.

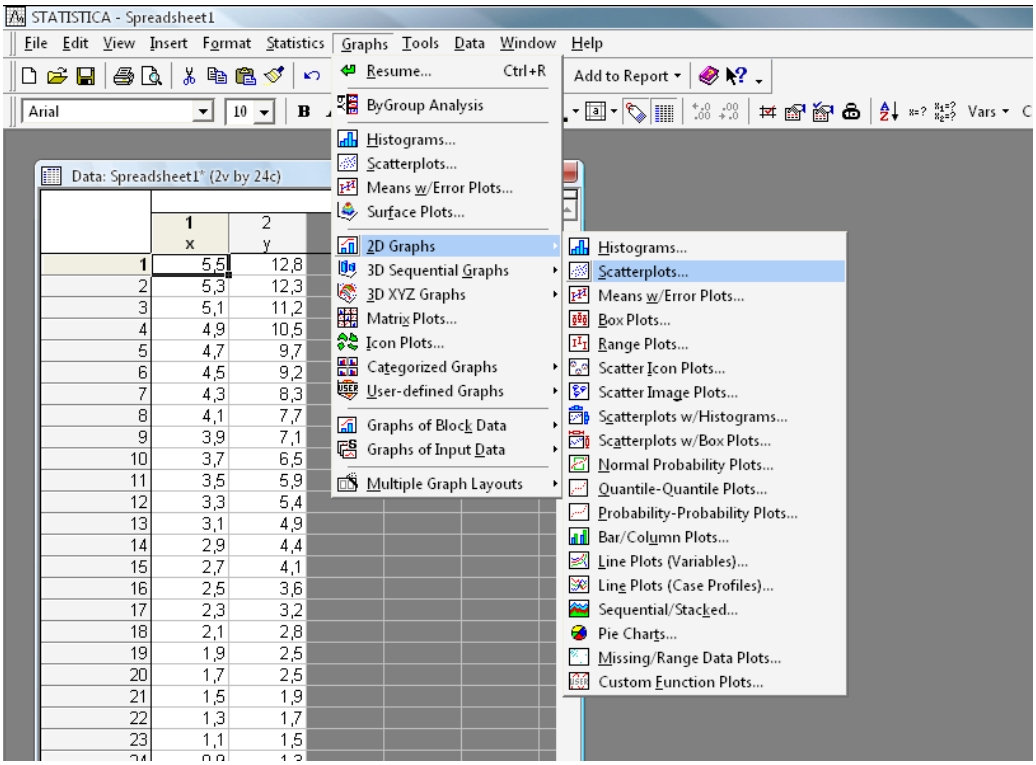



Рисунок 1.7 – Опції для побудови графіка залежності

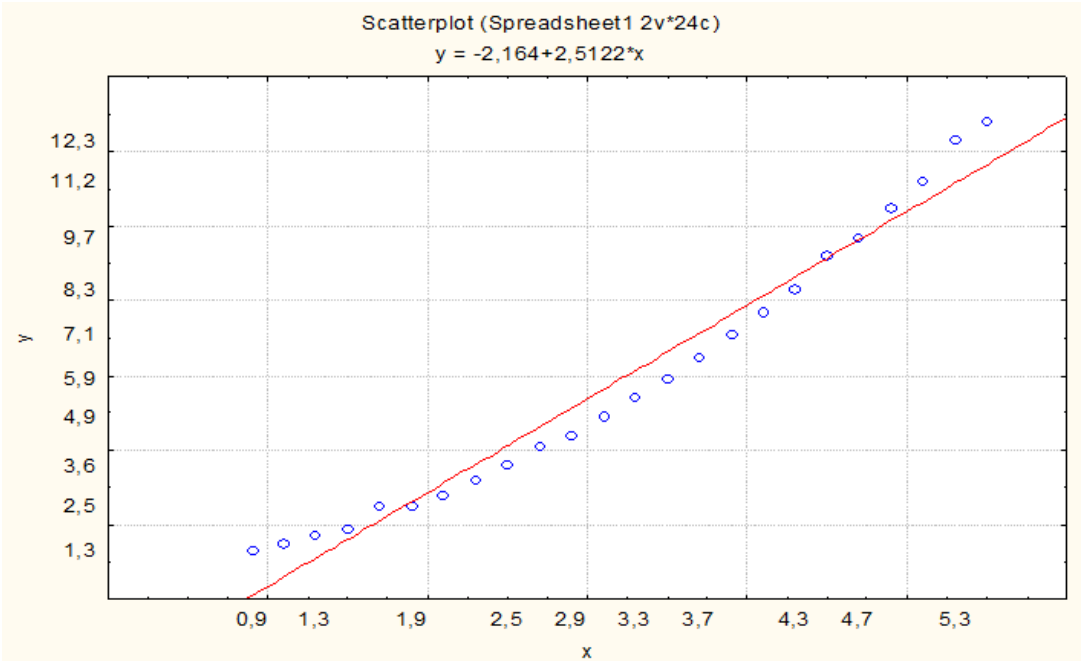



Рисунок 1.8 – Графік залежності $Y$ від $X$

Модель є нелінійною і для її побудови в пакеті Statistica необхідно застосувати модуль нелінійного оцінювання. Оберемо в меню: Statistic -> Advanced Linear/Nonlinear Models -> Nonlinear Estimation.

Далі обираємо User-specified regression, custom loss function (рис. 1.9) – регресія, що задана користувачем, і функція залишків.

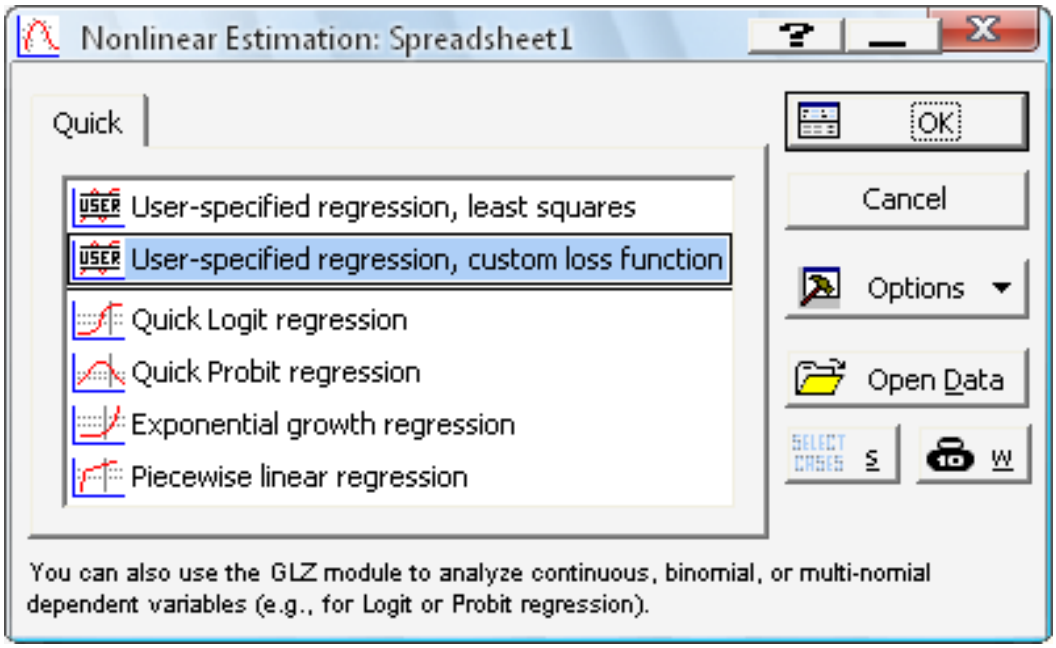



Рисунок 1.9 – Регресія, що задана користувачем, і функція залишків

В отриманому вікні обираємо тип рівняння (рис. 1.10) та натискаємо OK (рис. 1.11).

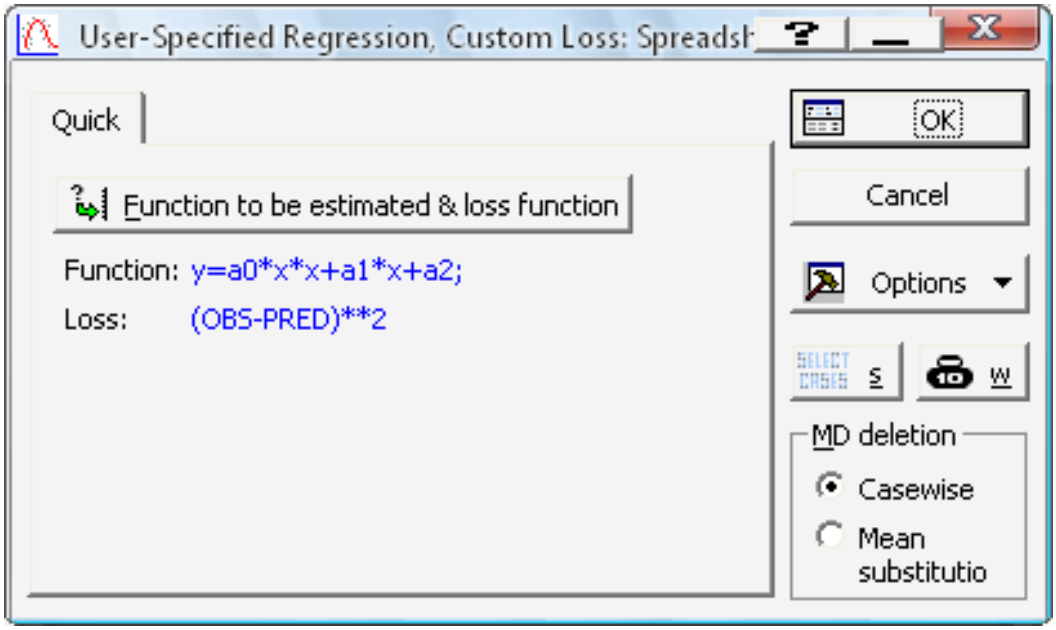



Рисунок 1.10 – Вибраний тип рівняння

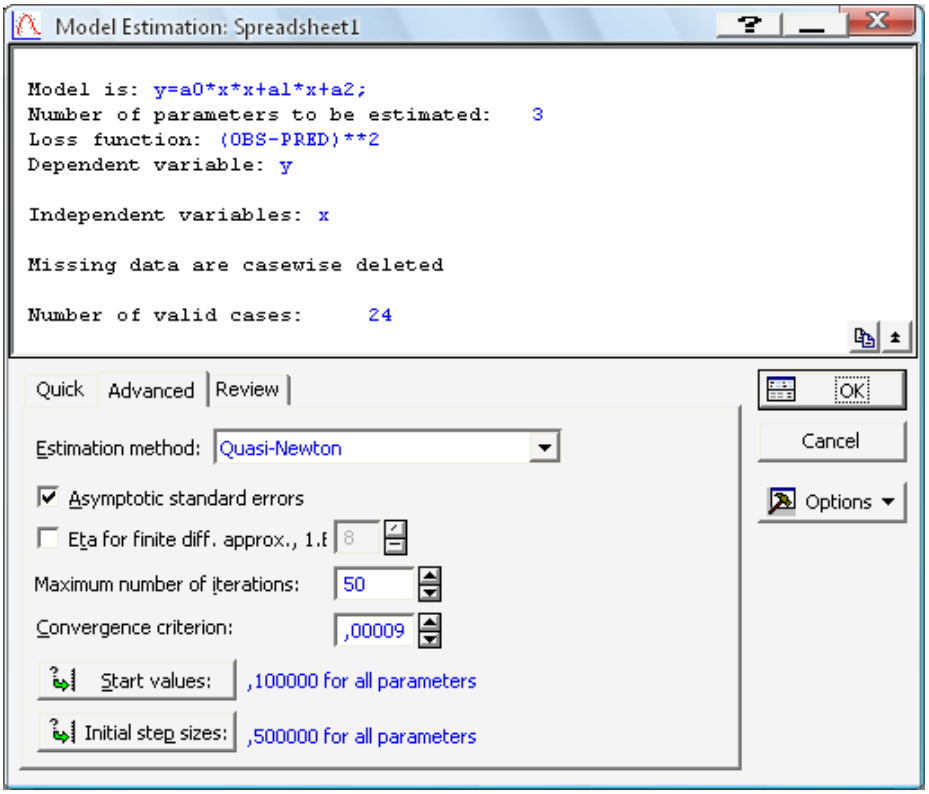



Рисунок 1.11 – Результат здійсненого вибору

У результаті отримаємо значення параметрів рівняння регресії (рис. 1.12).

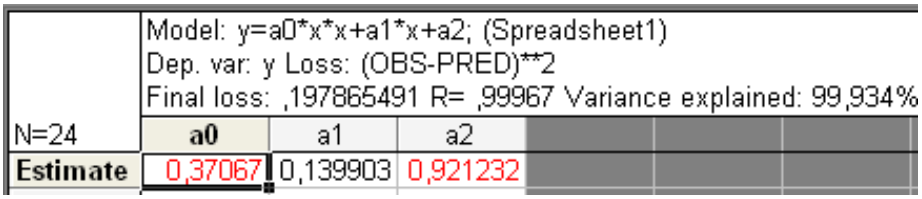



Рисунок 1.12 – Значення параметрів рівняння регресії

Рівняння має вигляд:

$$y = 0{,}37067\,x^2 + 0{,}139903\,x + 0{,}921232.$$

Що означають тут решта значень? Це залишкові дисперсії (похибки).

### 1.2.3 Завдання для індивідуального виконання роботи

За допомого датчика псевдовипадкових чисел сформувати масиви $(x_i, y_i)_{i=1}^{n}$ із заданою апріорно залежністю у вигляді $y_i = f(x_i) + \xi_i$, де $\xi_i$ – нормально розподілена випадкова величина.

Оцінити лінійну середньоквадратичну регресію $Y$ на $X$ та визначити коефіцієнт кореляції величин $X$, $Y$. Зробити висновок про зв'язок між $X$, $Y$.

Оцінити параметри $a, b, c$ параболічної кореляції другого порядку виду $y = ax^2 + bx + c$.

### 1.2.4 Контрольні запитання та завдання

1. Пояснити, що таке «коефіцієнт кореляції» і «регресія двовимірної випадкової величини».
2. Пояснити поняття «середньоквадратичної регресії».
3. Пояснити поняття «лінійної регресії» та «нелінійної регресії».
4. Як пов'язані між собою «вибірковий коефіцієнт кореляції» та «кореляційне відношення»?
5. Пояснити поняття «оптимальність» та застосування залишкової дисперсії (похибки).
6. Пояснити поняття «криволінійна кореляція».
7. Пояснити, яким чином приймається рішення про відповідність даних заданій моделі з використанням пакету Статистика.

### 1.3 Побудова правил класифікації та дерев рішення

Класифікаційні правила складаються з двох частин: умови та висновків:

*якщо* (умова), *то* (висновок).

Умовою є перевірка однієї або кількох незалежних змінних. Перевірки декількох змінних можуть бути об'єднані з допомогою операцій «і», «або» і «не».

Розглянемо (табл. 1.4) дані (навчальна вибірка) для оцінки можливості проведення футбольних матчів [1].

Таблиця 1.4 – Дані навчальної вибірки

| **Наблюдение** | **Температура** | **Влажность** | **Ветер** | **Игра** |
|---|---|---|---|---|
| Солнце | Жарко | Высокая | Нет | Нет |
| Солнце | Жарко | Высокая | Есть | Нет |
| Облачность | Жарко | Высокая | Нет | Да |
| Дождь | Норма | Высокая | Нет | Да |
| Дождь | Холодно | Норма | Нет | Да |
| Дождь | Холодно | Норма | Есть | Нет |
| Облачность | Холодно | Норма | Есть | Да |
| Солнце | Норма | Высокая | Нет | Нет |
| Солнце | Холодно | Норма | Нет | Да |
| Дождь | Норма | Норма | Нет | Да |
| Солнце | Норма | Норма | Есть | Да |
| Облачность | Норма | Высокая | Есть | Да |
| Облачность | Жарко | Норма | Нет | Да |
| Дождь | Норма | Высокая | Есть | Нет |

Висновком є значення залежної змінної або розподіл її імовірності по класах, наприклад:

*якщо* (спостереження = сонце і температура = жарко), *то* (гра = немає);

*якщо* (спостереження = хмарність і температура = холодно), *то* (гра = є).

Основними перевагами правил є легкість їх сприйняття та запис на природній мові, а також відносна їх незалежність. У набір правил легко додати нову інформацію, не змінюючи існуючу. Відносність незалежності правил пов'язана з можливою їх суперечливістю один одному. Якщо змінні, що характеризують об'єкт, задовольняють умовним частинам правил з різними висновками, то виникає невизначеність зі значенням його залежної змінної:

*якщо* (спостереження = сонце), *то* (гра = немає);

*якщо* (спостереження = хмарність і температура = холодно), *то* (гра = є).

*Дерево рішень* – це спосіб аналізу даних у вигляді ієрархічної послідовної структури.

Кожен вузол дерева включає перевірку певної незалежної змінної. Іноді у вузлі дерева дві незалежні змінні порівнюються одна з одною або визначається деяка функція від однієї чи декількох змінних.

Листя дерев відповідають значенням залежної змінної, тобто класам.

Об'єкт належить певному класу, якщо значення його незалежних змінних задовольняють умовам, записаним у вузлах дерева на шляху від кореня до листа, відповідному цього класу. Дерева рішень легко перетворюються у правила. Необхідно зауважити, що зворотне перетворення від правил до дерева не завжди можливо. Це пов'язано з більшою свободою запису правил. Наприклад, при використанні операції «або» у побудованому за таким правилом дереві виникне необхідність у дублюванні піддерев.

### 1.3.1 Теоретичні засади

Розглянемо найпростіший метод побудови *правил класифікації* [12], що будує правила за значенням однієї незалежної змінної, тому в літературі його часто називають «1-правило» (1-rule) або коротко правило ***1R***.

Ідея методу проста. Для будь-якого можливого значення кожної незалежної змінної формується правило, яке класифікує об'єкти з навчальної вибірки. У заключній частині правила вказується значення залежної змінної, яке найбільш часто зустрічається у об'єктів з обраним значенням незалежної змінної. У цьому випадку помилкою правила є кількість об'єктів, що мають те ж значення аналізованої змінної, але не відносяться до обраного класу.

Таким чином, для кожної змінної буде отримано набір правил (для кожного значення). Оцінивши ступінь помилки кожного набору, вибирається змінна, на якій побудовані правила з найменшою помилкою (табл. 1.5).

Таблиця 1.5 – Отримані правила і їх оцінки

| Правило | Помилка |
|---|---|
| Якщо (спостереження = сонце) то (гра = немає) | 2/5 |
| Якщо (спостереження = хмарно) то (гра = так) | 0/4 |
| Якщо (спостереження = дощ) то (гра = так) | 2/5 |
| Якщо (температура = жарко) то (гра = немає)* | 2/4 |
| Якщо (температура = норма) то (гра = так) | 2/6 |
| Якщо (температура = холодно) то (гра = так) | 1/4 |
| Якщо (вологість = висока) то (гра = немає) | 3/7 |
| Якщо (вологість = норма) то (гра = так) | 1/7 |
| Якщо (вітер = немає) то (гра = так) | 2/8 |
| Якщо (вітер = є) то (гра = немає)* | 3/6 |

Якщо у навчальній вибірці зустрічаються об’єкти з пропущеними значеннями незалежних змінних, то алгоритм 1R підраховує такі об’єкти для кожного можливого значення змінної.

Іншою проблемою для розглянутого методу є чисельні значення змінних. Очевидно, що якщо змінна має дійсний тип, то кількість можливих значень може бути нескінченною.

Для вирішення цієї проблеми всю область значень такої змінної розбивають на інтервали, щоб кожен з них відповідав визначеному класу у навчальній вибірці.

У результаті буде отримано набір дискретних значень, з якими може працювати даний метод.

Припустимо, що дані змінної температура, наведені в таблиці 1.4, мають такі числові значення та відповідні їм значення залежної змінної (табл. 1.6).

Таблиця 1.6 – Числові значення та відповідні їм значення залежної змінної навчальної вибірки (табл. 1.4)

| 4 | 5 | 8 9 10 | 11 12 | 12 15 15 | 20 | 21 23 | 25 |
|---|---|---|---|---|---|---|---|
| так | немає | так так так | ні ні | так так так | немає | так так | немає |

Діапазон значень можна розбити на інтервали таким чином:

{До 4,5; 4,5-7,5; 7,5-10,5; 10,5-12; 12-17,5; 17,5-20,5; 20,5-24; більше 24}.

Інша проблема методу – це надчутливість. Справа в тому, що він вибирає змінні, що приймають найбільшу кількість можливих значень, тому що для них помилка буде найменшою. Наприклад, для змінної, що є ключем (тобто для кожного об’єкта своє унікальне значення), помилка буде дорівнює нулю. Однак, для таких змінних правила будуть зайвими, тому при формуванні навчальної вибірки для даного методу важливо правильно вибрати набір незалежних змінних.

Метод 1R, незважаючи на свою простоту, у багатьох випадках на практиці виявляється досить ефективним. Це пояснюється тим, що більшість об’єктів дійсно можна класифікувати лише по одному атрибуту. Крім того, мала кількість правил дозволяє легко зрозуміти і використовувати отримані результати.

Часто для класифікації необхідно розглянути кілька незалежних змінних. Таку класифікацію дозволяє виконувати метод побудови правил класифікації ***Naive Bayes***, який використовує формулу Баєса для розрахунку імовірності. Назва Naive (наївний) походить від наївного припущення, що розглядаємо змінні, які вважаємо незалежними [12].

При використанні формули Баєса для оцінки достовірності правила виникає проблема, пов’язана з тим, що у навчальній вибірці може не бути жодного об’єкта, який має певне значення змінної та відноситься до певного класу. У цьому випадку імовірність буде дорівнювати 0, імовірність такого правила також дорівнює «0». Щоб уникнути цього, до кожної імовірності додається деяке значення, відмінне від нуля. Така методика називається оціночною функцією Лапласа.

Одним з дійсних переваг цього методу є те, що пропущені значення не створюють ніякої проблеми.

При підрахунку імовірності вони просто пропускаються для всіх правил, і це не впливає на співвідношення імовірностей.

Числові значення незалежних змінних, зазвичай, обробляються з урахуванням того, що вони мають нормальний розподіл імовірностей або розподіл Гауса [13]. Для них визначається математичне сподівання і середньоквадратичне відхилення [14].

Під час використання методів побудова дерева рішень буде відбуватися зверху вниз. Більшість з них є «жадібними алгоритмами». Це означає, що якщо один раз змінна була обрана і по ній було вироблено розбиття на підмножини, то алгоритм не може повернутися назад і вибрати іншу зміну, яка дала б краще розбиття. Тому на етапі побудови не можна сказати, чи дасть обрана змінна в кінцевому підсумку оптимальне розбиття.

Під час побудови дерев рішень ключова увага приділяється вибору змінної, по якій буде виконуватися розбиття.

Для побудови дерева на кожному внутрішньому вузлі необхідно знайти таку умову (перевірку), яка б розбивала множину, асоційовану з цим вузлом, на підмножини.

Повинна бути обрана одна з незалежних змінних. Загальне правило для вибору можна сформулювати таким чином: вибрана змінна повинна розбити множину так, щоб отримані у результаті підмножини складалися з об’єктів, що належать до одного класу, або були максимально наближені до нього, тобто кількість об’єктів з інших класів в кожній з цих множин була мінімальною.

До *методів побудови дерев рішень* відносять:

– ***метод ID3***. На початку роботи будується дерево з початковою множиною у кореневому вузлі. На кожній ітерації методу, він перебирає усі невикористані атрибути початкової множини та обчислює ентропію цих атрибутів. Обирається атрибут з найменшим значенням ентропії (або найбільшим інформаційним виграшем). Проводиться розбиття початкової множини за вибраним атрибутом для отримання підмножин даних (наприклад, менше 50, від 50 до 100, більше 100). Метод працює на кожній підмножині, враховуючи лише ті атрибути, які раніше не були вибрані. Він будує дерево рішень, де нетермінальні вузли (внутрішні вузли) представляють вибраний атрибут, а термінальні вузли (листові вузли) – мітку класу кінцевої підмножини цього вузла розгалуження;

– ***метод покриття*** (побудова дерев рішень для кожного класу окремо, на кожному етапі генерується перевірка вузла дерева, який покриває кілька об'єктів навчальної вибірки). При наявності у об'єктів тільки двох змінних їх можна представити у вигляді точок двомірного простору. Об'єкти, що відносяться до різних класів, позначаються знаками «+» і «–». Для побудови правил за допомогою даного методу в навчальній вибірці повинні бути присутніми всі комбінації значень незалежних змінних. На кожному кроці вибирається значення змінної, яке поділяє всю множину на дві підмножини. Поділ має здійснюватися так, щоб всі об'єкти класу, для якого будується дерево, належали одній підмножині. Таке розбиття проводиться до тих пір, поки не буде побудована підмножина, що містить тільки об'єкти одного класу. Після побудови дерева для одного класу таким же чином будуються дерева для інших класів.

Нехай кожний об'єкт $O_i$ характеризується набором змінних

$$O_i = (x_1,...,x_k, x_n, y), \tag{1.10}$$

де $x_k$ – незалежні;

$y$ – залежна змінна.

Приклад залежної змінної – гра буде чи ні. Якщо множина значень залежної змінної $y$ скінченна, тобто $y \in C_y, C_y = (c_1,...,c_r)$, то маємо задачу класифікації.

Завдання класифікації – це задача віднесення будь-якого об'єкта до одного з наперед відомих класів, наприклад, «Давати» / «Не давати кредит» (рис. 1.13).

На основі наявних апріорних даних будується дерево. При цьому клас кожної із ситуацій вважається відомим. При побудові дерева всі дані навчальної вибірки спочатку потрапляють у верхній вузол, а потім розподіляються по вузлах, які, у свою чергу, також можуть бути розбиті на дочірні вузли.

Як правило, кожний вузол дерева включає перевірку визначеної незалежної змінної [1].

Дерева рішень можна завжди перетворити в набір правил (навпаки – не завжди можливо із-за можливих протиріч у правилах).

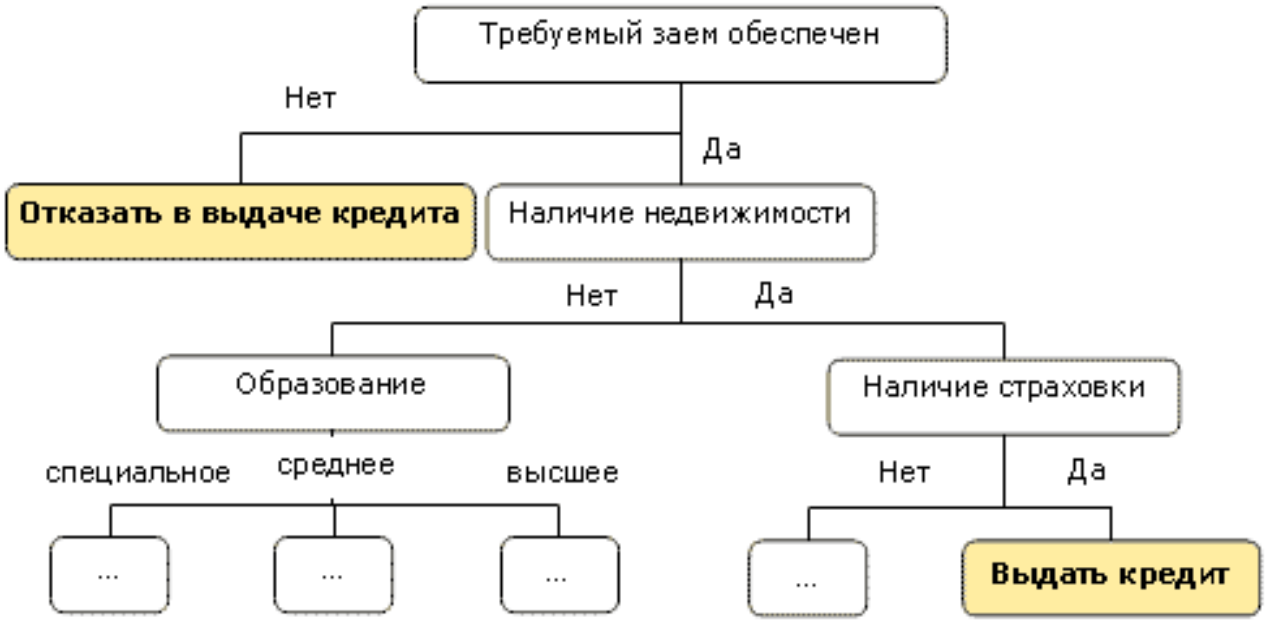



Рисунок 1.13 – Приклад дерева рішень при видачі кредиту

Основні проблеми побудови дерева рішень – вибір змінної для розбиття та проблема зупинки розбиття. Вони вирішуються, в основному, евристичним шляхом, причому, більшість правил застосовується тільки для часткових випадків. Треба віддавати перевагу побудові дерева, що складається із незначної кількості вузлів, яким відповідає велика кількість об’єктів навчальної вибірки.

При побудові дерев рішень основна увага приділяється наступним питанням: вибору критерію атрибуту, за яким піде розбиття, зупинки навчання і відсікання гілок.

*Правило побудови* (створення дерева рішень). Процес створення відбувається зверху вниз, тобто є низхідним. У ході процесу алгоритм повинен розбити множину на підмножини, які б асоціювалися з цим вузлом перевірки, таким чином, щоб об’єкти підмножин, отриманих у результаті цього розбиття, були представниками одного класу або ж були максимально наближені до такого розбиття.

*Правило зупинки*. Визначаємо, чи є даний вузол внутрішнім вузлом та буде розбиватися далі, або ж він є кінцевим вузлом, тобто вузлом рішення.

*Правило відсікання гілок*. Процес проходить знизу вгору, тобто є висхідним. Відсікання гілок слід проводити там, де ця процедура не призводить до зростання помилки, тобто кількість неправильно класифікованих об’єктів не збільшується.

Область застосування дерев рішень можна об’єднати в три класи:

1. Опис даних. Дерева рішень дають можливість зберігати інформацію про дані в компактній формі, що містить точний опис об’єктів.

2. Класифікація. Дерева рішень вирішують задачі класифікації, тобто віднесення об’єктів до одного з відомих класів. Цільова (залежна) змінна при цьому повинна мати дискретні значення.

3. Регресія. Цільова змінна має безперервні значення, дерева рішень встановлюють залежність цільової змінної від незалежних (вхідних) змінних. До цього класу належать задачі чисельного прогнозування.

### 1.3.2 Приклад реалізації алгоритму ID3

Для реалізації алгоритму побудови дерева рішень ID3 (Iterative Dichotomizer) візьмемо дані (табл. 1.7), що дозволяють рекомендувати тип контактних лінз [1].

На кожному кроці алгоритму обирається значення змінної, яке розділяє всю множину на дві підмножини. Поділ має здійснюватися так, щоб всі об’єкти класу, для якого будується дерево, належали одній підмножині. Таке розбиття проводиться до тих пір, поки не буде побудована підмножина, що містить тільки об’єкти одного класу.

Загальна ентропія (розрахунок однорідності вхідних даних):

$$Entropy(S) = -\frac{p}{p+n+z}\log_2\left(\frac{p}{p+n+z}\right) - \frac{n}{p+n+z}\log_2\left(\frac{n}{p+n+z}\right) - \frac{z}{p+n+z}\log_2\left(\frac{z}{p+n+z}\right), \quad (1.11)$$

де $p$, $n$ і $z$ – кількість об’єктів із множини $S$, що відносяться до одного і того ж класу. Повністю однорідні вхідні дані мають ентропію «0».

Розбиваємо тестові значення залежної змінної за рекомендаціями: («Ні» – 15/24, «М’які» – 5/24, «Жорсткі» – 4/24, 24 – загальна кількість значень).

Рахуємо загальну ентропію:

$$Entropy(15,5,4) = -(15/24)Log_2(15/24) - (5/24)Log_2(5/24) - (4/24)Log_2(4/24) = 1.3261.$$

*Entropy* не дорівнює «0», тобто вхідні дані необхідно розбити.

Знаходимо найбільш значуще розбиття серед незалежних змінних. Для кожного стовпця (незалежна змінна) підраховуємо ентропію, виходячи із її значень.

Таблиця 1.7 – Дані для реалізації алгоритму побудови дерева рішень

| Возраст | Предписание | Астигматизм | Степень износа | Рекомендации |
|---|---|---|---|---|
| Юный | Близорукость | Нет | Пониженный | Нет |
| Юный | Близорукость | Нет | Нормальный | Мягкие |
| Юный | Близорукость | Да | Пониженный | Нет |
| Юный | Близорукость | Да | Нормальный | Жесткие |
| Юный | Дальнозоркость | Нет | Пониженный | Нет |
| Юный | Дальнозоркость | Нет | Нормальный | Мягкие |
| Юный | Дальнозоркость | Да | Пониженный | Нет |
| Юный | Дальнозоркость | Да | Нормальный | Жесткие |
| Пожилой | Близорукость | Нет | Пониженный | Нет |
| Пожилой | Близорукость | Нет | Нормальный | Мягкие |
| Пожилой | Близорукость | Да | Пониженный | Нет |
| Пожилой | Близорукость | Да | Нормальный | Жесткие |
| Пожилой | Дальнозоркость | Нет | Пониженный | Нет |
| Пожилой | Дальнозоркость | Нет | Нормальный | Мягкие |
| Пожилой | Дальнозоркость | Да | Пониженный | Нет |
| Пожилой | Дальнозоркость | Да | Нормальный | Нет |
| Старческий | Близорукость | Нет | Пониженный | Нет |
| Старческий | Близорукость | Нет | Нормальный | Нет |
| Старческий | Близорукость | Да | Пониженный | Нет |
| Старческий | Близорукость | Да | Нормальный | Жесткие |
| Старческий | Дальнозоркость | Нет | Пониженный | Нет |
| Старческий | Дальнозоркость | Нет | Нормальный | Мягкие |
| Старческий | Дальнозоркость | Да | Пониженный | Нет |
| Старческий | Дальнозоркость | Да | Нормальный | Нет |

***1. Розбиваємо за віком.***

«Юний» – 8/24 («Ні» – 4/8, «М'які» – 2/8, «Жорсткі» – 2/8 (чисельник – кількість значень «Юний» серед вказаних значень залежної змінної)).

$$Entropy(4,2,2)=-(4/8)Log_2(4/8)-(2/8)Log_2(2/8)-(2/8)Log_2(2/8)=1,5.$$

«Літній» – 8/24 («Ні» – 4/8, «М'які» – 2/8, «Жорсткі» – 2/8).

$$Entropy(4,2,2)=-(4/8)Log_2(4/8)-(2/8)Log_2(2/8)-(2/8)Log_2(2/8)=1,3.$$

«Старечий» – 8/24 («Ні» – 4/8, «М'які» – 2/8, «Жорсткі» – 2/8).

$$Entropy(4,2,2)=-(4/8)Log_2(4/8)-(2/8)Log_2(2/8)-(2/8)Log_2(2/8)=1,1.$$

Значимість розбиття:

$$Gain(A) = E(загальна) - \sum a \cdot E(нащадків), \qquad (1.12)$$

де $a$ – ваговий коефіцієнт.

$$Gain(Вік)=1.3261-(8/24*E(Юний)+8/24*E(Літній)+8/24*E(Старечий))=0,0394$$

***2. Розбиваємо за приписом.***

«Короткозорість» – 12/24 («Ні» – 7/12, «М'які» – 2/12, «Жорсткі» – 3/12).

$$Entropy(4,2,2)=-(7/12)Log_2(7/12)-(2/12)Log_2(2/12)-(3/12)Log_2(3/12)=1,4.$$

«Далекозорість» – 12/24 («Ні» – 8/12, «М'які» – 3/12, «Жорсткі» – 1/12).

$$Entropy(8,3,1)=-(8/12)Log_2(8/12)-(3/12)Log_2(3/12)-(1/12)Log_2(1/12)=1,2.$$

Значимість розбиття:

$$Gain(Припис)=1.3261-(12/24*E(Короткозор)+12/24*E(Далекозор))=0,0394.$$

***3. Розбиваємо за астигматизмом.***

«Наявний» – 12/24 («Ні» – 8/12, «М'які» – 0/12, «Жорсткі» – 4/12).

$$Entropy(4,2,2)=-(7/12)Log_2(7/12)-(2/12)Log_2(2/12)-(3/12)Log_2(3/12)=0,9.$$

«Відсутній» – 12/24 («Ні» – 7/12, «М'які» – 5/12, «Жорсткі» – 0/12).

$$Entropy(8,3,1)=-(8/12)Log_2(8/12)-(3/12)Log_2(3/12)-(1/12)Log_2(1/12)=0,4.$$

Значимість розбиття:

$$Gain(Астигматизм)=1.3261-(12/24*E(Наявний)+12/24*E(Відсутній))=0,3770.$$

Приклад фрагмента програми. Кількість рекомендацій людей, у яких присутній астигматизм, визначаємо так:

```
     Foreach (Person per in
(IList<Person>)this.dataGridView1.DataSource)
     {
         If (per.Astigmatism.Equals(Astigmatism.Yes))
         {
             If (per.Result.Equals(Result.No)) //немає рекомендацій
             {
                c1++;
             }
             If (per.Result.Equals(Result.Hard)) //жорсткі рекомендації
             {
                 c3++;
             }
             If (per.Result.Equals(Result.Soft)) //м'які рекомендації
             {
                 c2++;
             }
         }
     }
```

***4. Розбиваємо по ступеню зносу.***

«Нормальний» – 12/24 («Ні» – 3/12, «М'які» – 5/12, «Жорсткі» – 4/12).

$$Entropy(3,5,2)=-(3/12)Log_2(3/12)-(5/12)Log_2(5/12)-(4/12)Log_2(4/12)=0,55.$$

«Знижений» – 12/24 («Ні» – 12/12, «М'які» – 0/12, «Жорсткі» – 0/12).

$$Entropy(12,0,0)=-(12/12)Log_2(12/12)-(0/12)Log_2(0/12)-(0/12)Log_2(0/12)=0.$$

Ентропія дорівнює «0», тобто побудована підмножина повністю однорідна (об’єкти відносяться до одного класу). Розбиття підмножин більше не потрібно.

Значимість розбиття:

$$Gain(Ступ\ зносу)=1.3261-(12/24*E(Нормальний)+ 12/24*E(Знижений))=0,5488.$$

Виходячи з критеріїв *Gain* для всіх незалежних змінних, вибираємо змінну з максимальним значенням: $MAX\{Gain\} = 0,5488$.

Розбиття по ступеню зносу є найбільш значимим. Тому спочатку розбиваємо за цією ознакою. Наступні за значимістю розбиття по наявності астигматизму.

Значимість розбиття для кожного параметра:

```
List<double> Gain = new List<double>();
Gain.Add(gainWearout);
Gain.Add(gainAge);
Gain.Add(gainSickness);
Gain.Add(gainAstigmatism);
Gain.Sort();
static String nodeFormat = "{0}: {1}";
int k = 1;//рівень вкладеності
```

Приєднання значення до відсортованого списку (дерева), починаючи з останнього елемента:

```
for (int i = Gain.Count - 1; i > -1; i--)
{
    if (k == 1) //додаємо у перший рівень дерева
    {
        if (Gain[i].Equals(gainWearout))

this.treeView1.Nodes.Add(String.Format(nodeFormat, Wearout.Name,
wearout));
        if (Gain[i].Equals(gainAge))

this.treeView1.Nodes.Add(String.Format(nodeFormat, Age.Name,
age));
        if (Gain[i].Equals(gainSickness))
```

```
this.treeView1.Nodes.Add(String.Format(nodeFormat,
Sickness.Name, sickness));
            if (Gain[i].Equals(gainAstigmatism))

this.treeView1.Nodes.Add(String.Format(nodeFormat,
Astigmatism.Name, astigmatism));
        }
    }
```

Додавання до дерева рекомендації і його розкриття:

```
    this.treeView1.Nodes[0].Nodes[0].Nodes[0].Nodes[0].Nodes.Ad
d(String.Format(nodeFormat, Result.Name, result));
    this.treeView1.ExpandAll();
```

Ступінь зносу: Нормальна
Астигматизм: Так
Вік: Юний
Рекомендації: Жорсткі
Вік: Літній
Припис: Далекозорість
Рекомендації: Ні
Припис: Короткозорість
Рекомендації: Жорсткі
Вік: Старечий
Припис: Далекозорість
Рекомендації: Ні
Припис: Короткозорість
Рекомендації: Жорсткі
Астигматизм: Ні
Вік: Юний
Рекомендації: М’які
Вік: Літній
Рекомендації: М’які
Вік: Старечий
Припис: Далекозорість
Рекомендації: М’які
Припис: Короткозорість
Рекомендації: Ні
Ступінь зносу: Знижена
Рекомендації: Ні

Отримаємо результат поданий на рисунку 1.14.

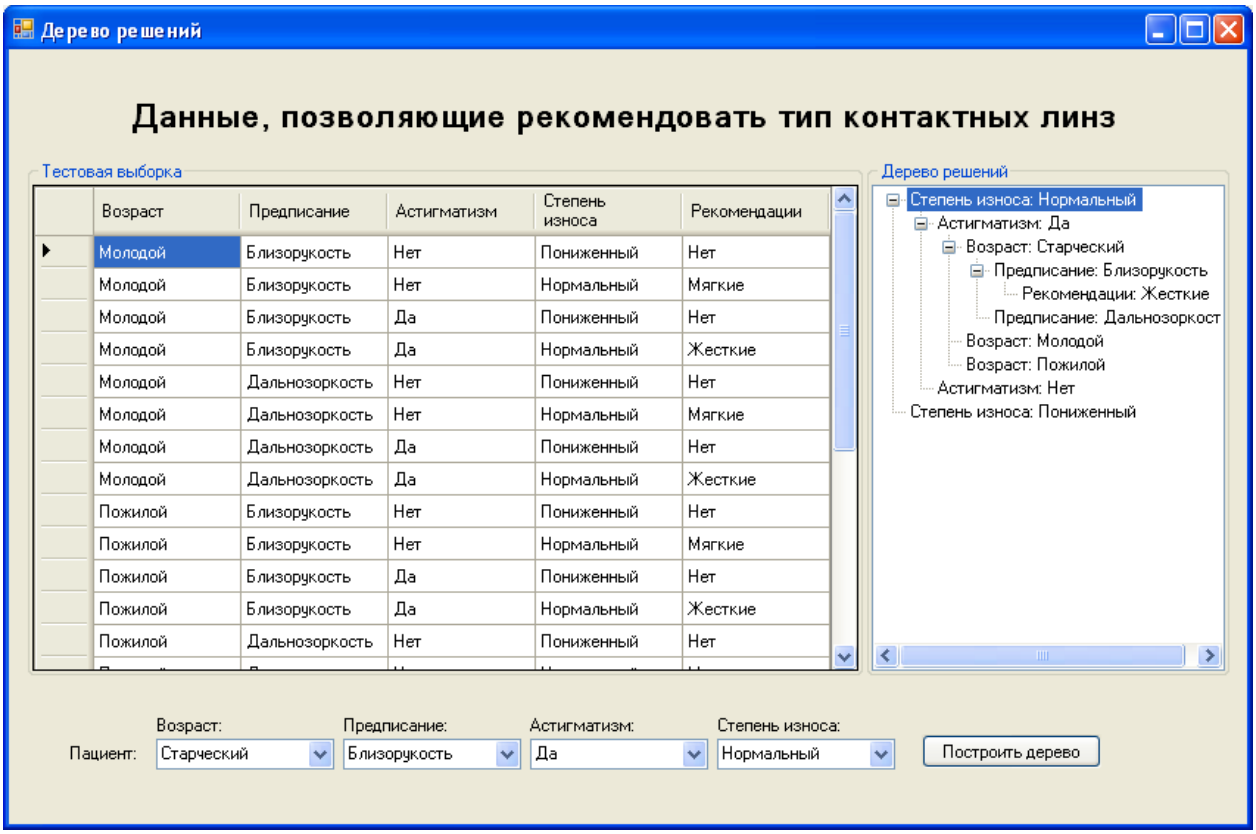



Рисунок 1.14 – Результати проведеного аналізу

### 1.3.3 Завдання для індивідуального виконання роботи

На основі таблиці навчальної вибірки (табл. 1.4) або запропонованої власної навчальної вибірки з метою вирішення конкретної практичної задачі, виконати наступне:

1) сформувати набір правил класифікації для множини значень залежної змінної;

2) побудувати дерево рішення для задачі класифікації;

3) застосувати правило 1R для формування набору правил класифікації;

4) застосувати алгоритм ID3 для побудови дерева рішень;

5) застосувати алгоритм покриття для побудови дерева рішень;

6) порівняти результати застосування евристик та методів формування правил класифікації і побудови дерев рішень для однієї і тієї ж задачі.

### 1.3.4 Контрольні запитання та завдання

1. Пояснити у чому полягає принципова відмінність між «класифікацією» та «регресією».
2. Пояснити поняття «класифікації», навести приклади практичних задач.
3. Пояснити суть підходу 1R для формування набору правил класифікації.
4. Пояснити суть підходу Naive Bayes для формування набору правил класифікації.
5. Пояснити, як пов’язані між собою «правила класифікації» та «дерева рішення».
6. Пояснити ідею алгоритма ID3 для побудови дерева рішень.
7. Пояснити у чому суть «алгоритма покриття» для побудови дерева рішень.

### 1.4 Формування асоціативних правил

Алгоритми побудови асоціативних правил вивчають можливі зв'язки чи комбінації об'єктів, подій, умов або висновків, оцінюють для них підтримку та формують правило прийняття рішень на цій основі.

Базовим поняттям є *транзакція* – множина подій, що відбуваються сумісно.

Число можливих асоціацій зі збільшенням числа об'єктів зростає експоненційно. Якщо в базі даних присутні $k$ об'єктів і всі асоціації є бінарними (містять по одному предмету в умові та висновку), то необхідно проаналізувати $k \cdot 2^{k-1}$ асоціацій.

Побудова такої кількості правил і обчислення імовірності кожного з них – задача складна. Для її вирішення використовуються алгоритмічні прийоми, що скорочують простір можливих рішень.

Однією з найпоширеніших є методика, заснована на виявленні так званих частих наборів, коли аналізуються тільки ті асоціації, які зустрічаються досить часто. На цій концепції заснований алгоритм ***Apriori***.

#### 1.4.1 Теоретичні засади

Однією із найбільш поширених задач ІАД є пошук наборів об'єктів із усієї множини об'єктів $I = \{i_j\}_{j=1}^{n}$, що часто зустрічаються разом.

Набори (підмножини) об'єктів із множини $I$ називаються транзакціями $T = \{i_j | i_j \in I\}$ [1]. Набір транзакцій формує деяку множину $D = \{T_1,....,T_m\}$, $m$ – число транзакцій. Множини $T_k$ можуть мати спільні елементи.

Відношення числа транзакцій, до яких входить набір $F$, до загального числа транзакцій, називається підтримкою $\sup pF$ набору $F$ і обчислюється як $\sup pF = |D_f| / |D|$, де модуль означає кількість елементів. При побудові асоціативних правил відбирають всі набори, що зустрічаються досить часто, тобто для яких $\sup pF \geq 0{,}5$.

Предметом сиквенційного аналізу є виявлення значимих послідовностей об'єктів, як правило, за часом аналізу або виникнення. Кажуть, що транзакція $T$ включає послідовність $S$, якщо $S \subseteq T$, і об'єкти, що входять до $S$, входять і до $T$ зі збереженням порядку. При цьому допускається, що між об'єктами із $S$ можуть знаходитися інші об'єкти.

Аналізується ієрархічна структура об’єктів, є можливість визначити зв’язки між об’єктами.

Поняття «покращення» показує, наскільки введене правило на основі асоціації корисливіше від правила випадкового вибору. Якщо «покращення» більше 1, це означає, що застосування правила має більший сенс, ніж випадковий вибір.

Алгоритм Apriori [1] реалізує процедуру виявлення частих наборів об’єктів.

### 1.4.2 Приклад реалізації асоціативного аналізу

Реалізуємо алгоритм Apriori для виявлення значимих наборів об’єктів із заданою величиною підтримки для таблиць навчальної вибірки, які можна запропонувати самостійно з метою вирішення конкретної практичної задачі. Роботу можна виконати шляхом побудови в MS Excel таблиць та обчислення значення підтримки відповідних наборів. Підтримка показує, який відсоток транзакцій підтримує дане правило.

Послідовність дій така:

– побудуємо таблицю 1.15 з транзакціями (набори куплених товарів);

– обчислюємо підтримку одиночних товарів;

– на основі значимих одиночних товарів шляхом перебору всіх можливих варіантів отримуємо таблицю з використанням пар товарів, у яких підтримка більше або дорівнює 0,5;

– на основі значимих одиночних та пар товарів отримуємо таблицю для трійок товарів, формуємо трійки, що мають значиму підтримку.

| A | B | C | D |
|---|---|---|---|
| 1 | 2 | 5 | 2 |
| 3 | 3 | 2 | 5 |
| 4 | 5 | 1 | 2 |
| | | 2 | |
| | | 3 | |

Рисунок 1.15 – Вхідні дані наборів (транзакцій)

Результатом роботи є сформовані набори із одиночних товарів (рис. 1.16), пар (рис. 1.17) та трійок (рис. 1.18) товарів.

| F | G | H |
|---|---|---|
| 1 | 0,5 | 1 |
| 2 | 0,75 | 2 |
| 3 | 0,75 | 3 |
| 4 | 0,25 | 5 |
| 5 | 0,75 | |

Рисунок 1.16 – Отримання наборів із одного елементу,
які мають підтримку 0,5 та вище (виділено)

| I | J | K |
|---|---|---|
| 1,2 | 0,25 | 1,3 |
| 1,3 | 0,5 | 2,3 |
| 1,5 | 0,25 | 2,5 |
| 2,3 | 0,5 | |
| 2,5 | 0,75 | |

Рисунок 1.17 – Отримання значимих пар елементів (виділено)

| L | M | N |
|---|---|---|
| 1,3,5 | 0,25 | 2,3,5 |
| 2,3,1 | 0,25 | |
| 2,3,5 | 0,5 | |
| 2,5,1 | 0,25 | |

Рисунок 1.18 – Формування значимих трійок (виділено)

Асоціативний аналіз дав можливість на основі множини транзакцій отримати набори із одиночних товарів, пар та трійок товарів, які із врахуванням зв'язків товарів мають значиму підтримку, що дає можливість продавати їх сумісно.

### 1.4.3 Завдання для індивідуального виконання роботи

Реалізувати алгоритм Apriori для виявлення наборів об'єктів із аналізом величини підтримки для різних таблиць навчальної вибірки, які можна запропонувати самостійно для вирішення практичної задачі. Допускається довільне формування навчальної вибірки або її формування в діалоговому режимі:

1) задати таблицю програмно з використанням випадкових чисел;
2) задати таблицю з попарно зв'язаними об'єктами;
3) задати таблицю з об'єктами, зв'язаними між собою у виді трійок.

### 1.4.4 Контрольні запитання та завдання

1. Пояснити поняття «транзакція» для множини об’єктів.
2. Пояснити поняття «асоціація».
3. У чому полягає суть сиквенційного аналізу?
4. Пояснити поняття «ієрархічне подання об’єкта» через «набори асоціацій».
5. Пояснити суть підтримки та «покращення».
6. У чому полягає теоретичне підґрунтя алгоритму Apriori?

### 1.5 Застосування кластерного аналізу

*Кластерний аналіз* – задача розбиття заданої вибірки об'єктів на підмножини, які називаються кластерами, так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися.

Кластеризацію відносять до статистичної обробки даних, а також до задач навчання без вчителя. У кластери поміщають групи елементів, які утворюються ґрунтуючись на відстані між ними. Кластеризація може бути сформульована як задача багатокритеріальної оптимізації. Результат кластеризації залежить від ряду факторів: вибраного методу, значень даних, числа сформованих кластерів, виду метрики, що застосовується для порівняння окремих елементів та кластерів між собою.

Ключовим моментом у кластерному аналізі є вибір метрики (міри схожості, відстані, близькості між двома об'єктами), від чого вирішальним чином залежить остаточний варіант розбиття об'єктів на групи при заданому методі.

*Агломеративні методи* ієрархічної кластеризації реалізують ідею «знизу верх» [10]. На початку кожна точка розглядається як кластер, потім метод намагається об'єднати найближчі сусідні точки в один більший кластер і так далі, щоб зрештою об'єднати всі кластери в один великий кластер. Ієрархічні агломеративні методи розрізняються за правилами побудови кластерів (одиничного зв'язку, повного зв'язку, середнього зв'язку та метод Варда та ін.), кожне з яких породжує специфічний метод.

Ієрархічні *дивізимні* (подільні) *методи* є логічною протилежністю агломеративних. Спочатку всі об'єкти належать одному кластеру, який на наступних кроках ділиться на менші кластери, у результаті утворюється послідовність груп.

#### 1.5.1 Теоретичні засади

Задачею кластеризації в ІАД є розбиття всієї множини об'єктів $I = \{i_j\}_{j=1}^{n}$, де $n$ – їх загальна кількість, на групи (кластери) шляхом побудови відображення $I \to C = \{c_j\}_{j=1}^{q}$ ($q$ – заздалегідь відома або невизначена кількість кластерів). У загальному плані при виконанні кластеризації отримані кластери можуть перетинатися (мати спільні елементи, тобто утворювати покриття множини $I$), або не мати спільних елементів (розбиття $I$) [15].

Для реалізації кластеризації необхідно задати міру $d(i_j, i_p)$ близькості між описами об'єктів, що у більшості випадків є метрикою $r(i_j, i_p)$ [1, 9].

Ключовим моментом у кластерному аналізі є вибір метрики, від якого залежить кінцевий варіант розбиття об'єктів.

Прикладами метрик є: Евклідова відстань, відстань Чебишева, відстань за Хеммінгом.

Застосування метрик у порівнянні з мірами близькості має перевагу через те, що при пошуку об'єктів у об'ємних базах даних можна використовувати внутрішні відстані, що скорочує час пошуку.

Більшість відомих алгоритмів використовують матрицю відмінностей $D = \{d(c_i, c_j)\}_{i,j=1}^{n}$, що визначає всі можливі відстані для об'єктів, що аналізуються.

У традиційних задачах кластеризації об'єкти мають вигляд числових векторів $i_p \in R^m$ із простору $R^m$.

Розглянемо найбільш поширені у застосуванні відстані та міри близькості, що характеризують взаємне розташування окремих груп об'єктів.

Нехай $c_j$ – $i$-та група (кластер) об'єктів, $m_j$ – кількість об'єктів, що утворюють групу $c_j$, вектор $w_j$ – середнє арифметичне кількості об'єктів, що входять до складу $c_j$, а $d(c_1, c_k)$ – відстань між групами $c_1$ та $c_k$ (рис. 1.19).

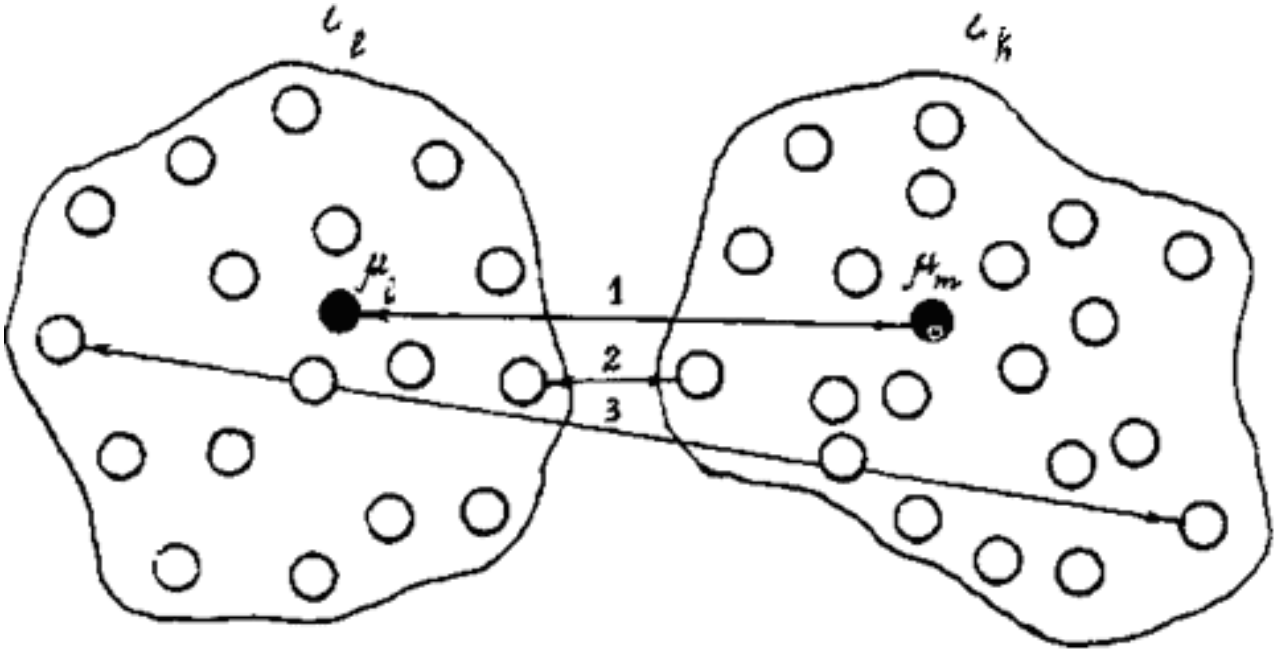



Рисунок 1.19 – Способи визначення відстаней між кластерами

На рисунку 1.19 цифрою «1» позначено метод визначення відстані між кластерами за центрами ваги, цифрою «2» – за найближчими сусідами, цифрою «3» – за найбільш віддаленими об’єктами.

Відстань найближчого сусіда визначається за формулою: $d_{\min}(c_1, c_k) = \min\limits_{\substack{i_a \in c_1 \\ i_b \in c_k}} d(i_a, i_b)$. Відстань найбільш віддаленого сусіда визначається за формулою $d_{\max}(c_1, c_k) = \max\limits_{\substack{i_a \in c_1 \\ i_b \in c_k}} d(i_a, i_b)$. Відстань між центрами ваги – це відстань між центральними точками кластерів: $d(c_1, c_m) = d(\mu_1, \mu_m)$.Розглянемо деякі конкретні способи формування кластерів.

1. Ієрархічна класифікація. Реалізує розбиття вибірки $\{i_j\}$, $i_j \in I$ на групи, що не перетинаються, та заснована на метриці $\rho(c_1, c_2)$ між кластерами $c_1 \subset C, c_2 \subset C$.

На кожному етапі оброблення поєднуються дані із двох найближчих за відстанню кластерів. Для прискорення обчислень чи завершення кластерування у процедурах ієрархічної класифікації застосовують систему порогів, яка поєднує тільки ті класи, міра близькості яких не перевищує значення порогу.

2. Групування на основі просіювання. Проводиться парне порівняння об’єктів $i_j$ з вибірки $\{i_j\}$. Для кожного об’єкту $i_j$ формується кортеж $i_{N(j)} = \{i_{j1}, ..., i_{jN}\}$ «схожих» об’єктів. Далі визначається сумарна віддаленість елементів кортежу $d(i_{N(j)}, i_j)$ від $i_j$. Вибірка $\{i_j\}$ впорядковується. Вважається, що $i_j \le i_k$, якщо

$$d(i_{N(j)}, i_j) \le d(i_{N(k)}, i_k). \qquad (1.13)$$

Класифікація за принципом просіювання здійснюється наступним чином: перший об’єкт з упорядкованого списку разом із своїм кортежем утворюють перший кластер, елементи вибірки, які не були видалені раніше, видаляються, процедура класифікації продовжується, доки уся вибірка або її зазначена доля не буде класифікованою.

### 1.5.2 Приклади реалізації методу кластеризації

**Приклад 1**. Розглянемо метод кластеризації на основі просіювання, реалізований у середовищі MATLAB.

Вхідними даними нехай будуть оцінки із діапазону 60–100.

Наведемо фрагмент програми у середовищі MATLAB.

*Задамо кількість елементів вхідної вибірки та саму вибірку.*

```
>>n=15;
>>M=[65; 92; 90; 70; 69; 60; 93; 99; 87; 68; 74; 89; 59;
83; 63];
```

*Сформуємо нульову матрицю nxn.*

```
>>kort=zeros(int32(n),int32(n));
```

*Заповнимо цю матрицю відстанями між елементами.*

```
>>for i1=1:1:n
    for i2=1:1:n
    if(i1~=i2)&(abs(M(i1)-M(i2))<5)
    kort(i1,i2)=abs(M(i1)-M(i2));
    end;
    end;
    end;
```

*Відсортуємо матрицю за сумою рядків.*

```
>>sortrows(kort);
```

*Відбираємо найбільш близько розташовані кортежі та формуємо з них окремий кластер.*

```
>>for i1=1:1:n
    for j1=1:1:n
    if(kort(i1,j1)~=0)
    for i2=i1+1:1:n
    kort(i2,j1)=0;
    kort(i2,i1)=0;
    end;
    end;
    end;
    end;
```

*Виводимо результат кластеризації у графічній формі.*

```
>>S=['0';'X';'+';'*';'S';'D';'V';'A';'<';'>';'P';'H'];
>>for i1=1:1:n
    for i2=1:1:n
    if(kort(i1,i2)~=0)
    plot(M(i2),S(rem(i1,12)+1))
    hold on
    end;
    end;
```

Зауважимо, що результат кластеризації залежить від порога, який у даному прикладі дорівнює значенню «5».

Результат виконання програми показано на рисунку 1.20.

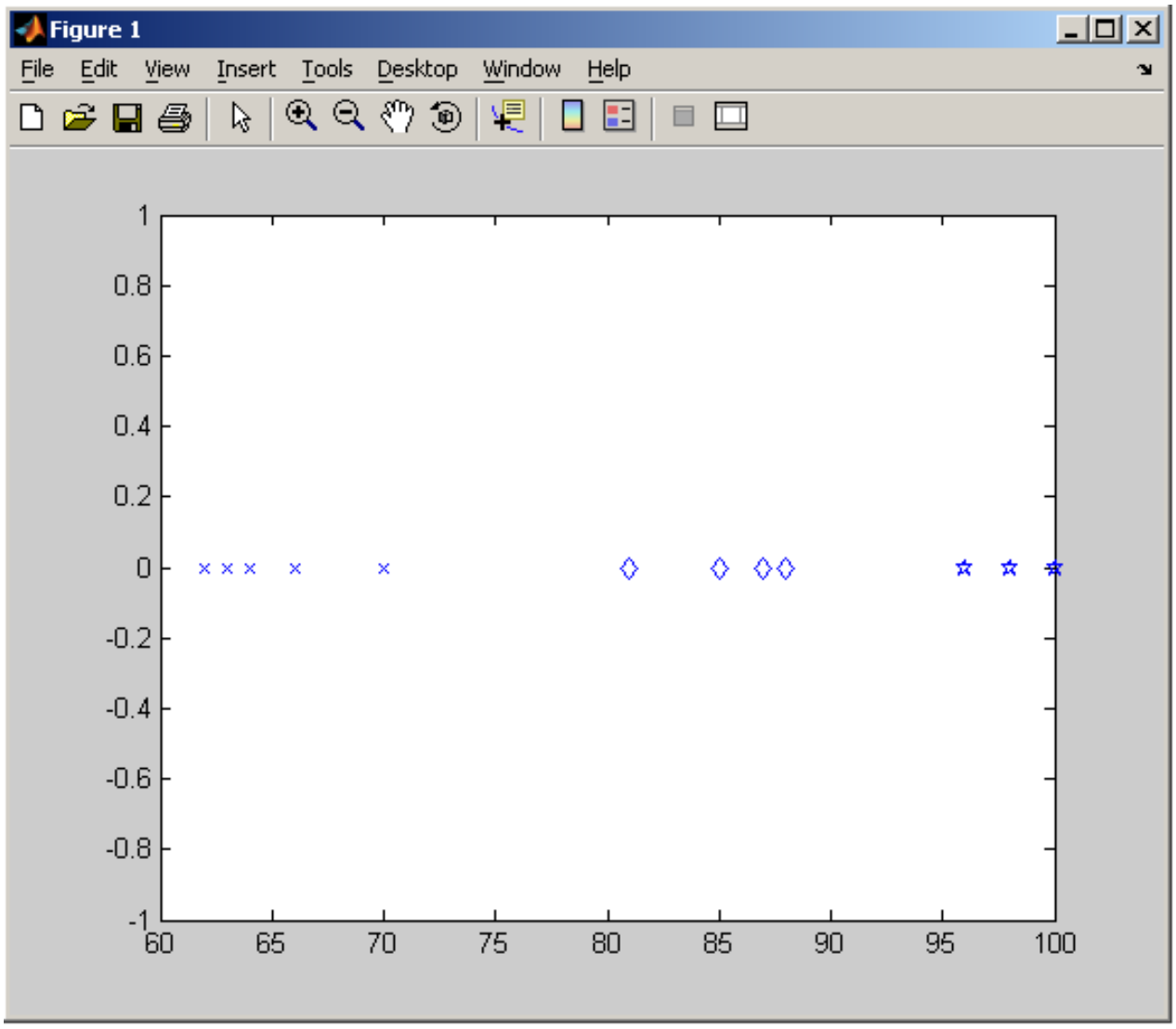



Рисунок 1.20 – Результат кластеризації методом просіювання

**Приклад 2.** Розглянемо кластеризацію на основі методу $k$-середніх, реалізовану за допомогою засобів мови програмування Python.

Вхідними даними нехай буде двовимірний вектор – кількість викладачів та кількість студентів у 200 закладах вищої освіти України (рис. 1.21).

На рисунку 1.21 наведено фрагмент початкових даних за таким шаблоном: «кількість студентів, кількість викладачів». База даних налічує 200 записів.

univers.csv

```
25000,3075
22000,3200
15000,2000
11500,1700
30000,2600
31000,4700
11380,1420
10000,640
16230,3000
21072,1610
8000,1100
3936,912
10010,763
10800,1570
15720,1030
11000,1300
21000,1830
10680,1701
16400,2100
17538,1230
16000,3470
5767,548
8500,1180
15400,2060
6000,780
7937,1830
4600,720
3155,647
2462,330
8172,1700
9797,790
10120,1016
3500,300
```

Рисунок 1.21 – Фрагмент вхідних даних

Вибірка початкових даних подана на рисунку 1.22.

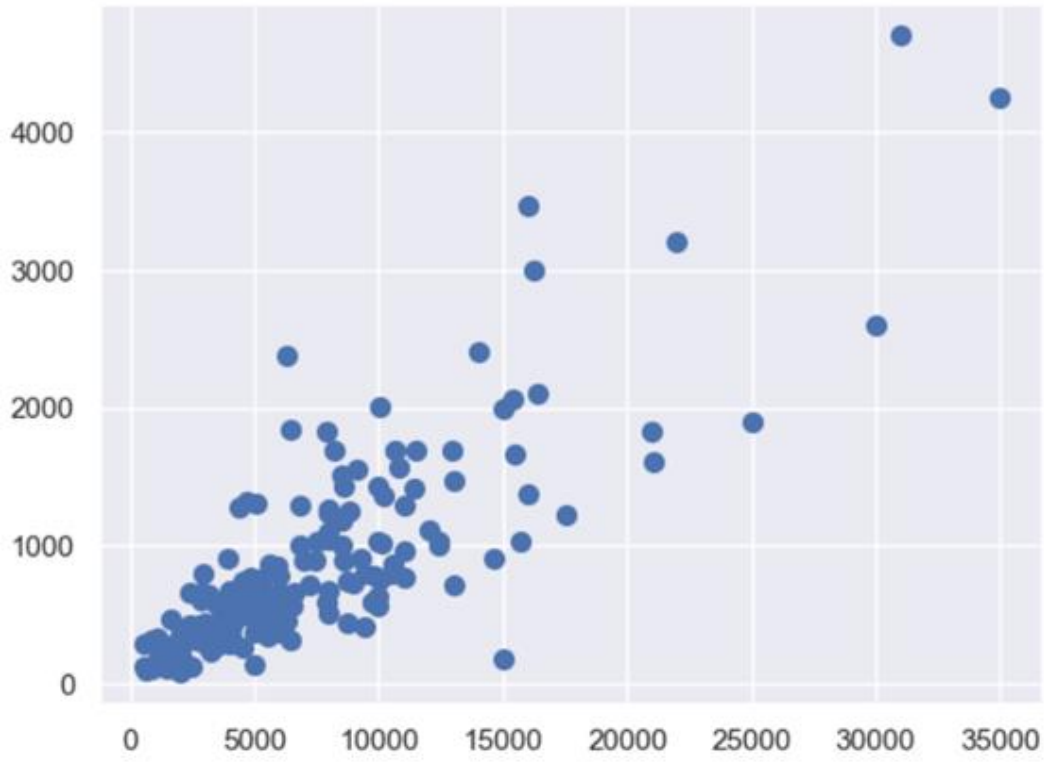



Рисунок 1.22 – Вхідні дані (некластеризована вибірка)

Наведемо вигляд програмного коду реалізації запропонованої задачі.

```
def find_clusters(X, n_clusters, rseed=2):
    # Вибираємо точки випадковим чином
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    while True:
        # Присвоюємо точки кожному найближчому центру
        labels = pairwise_distances_argmin(X, centers)
        # Перерахуємо центри, беручи середнє значення координат усіх точок,
          які належать до цього центру
        new_centers = np.array([X[labels == i].mean(0)
                                for i in range(n_clusters)])
        # Якщо координати центрів не змінились у порівнянні з попередньою
          ітерацією, то кластеризація закінчена
        if np.all(centers == new_centers):
            bresk
        centers = new_centers
    return centers, labels
```

Результат застосування кластерного аналізу показано на рисунку 1.23 (для трьох кластерів) та на рисунку 1.24 (для п’яти кластерів).

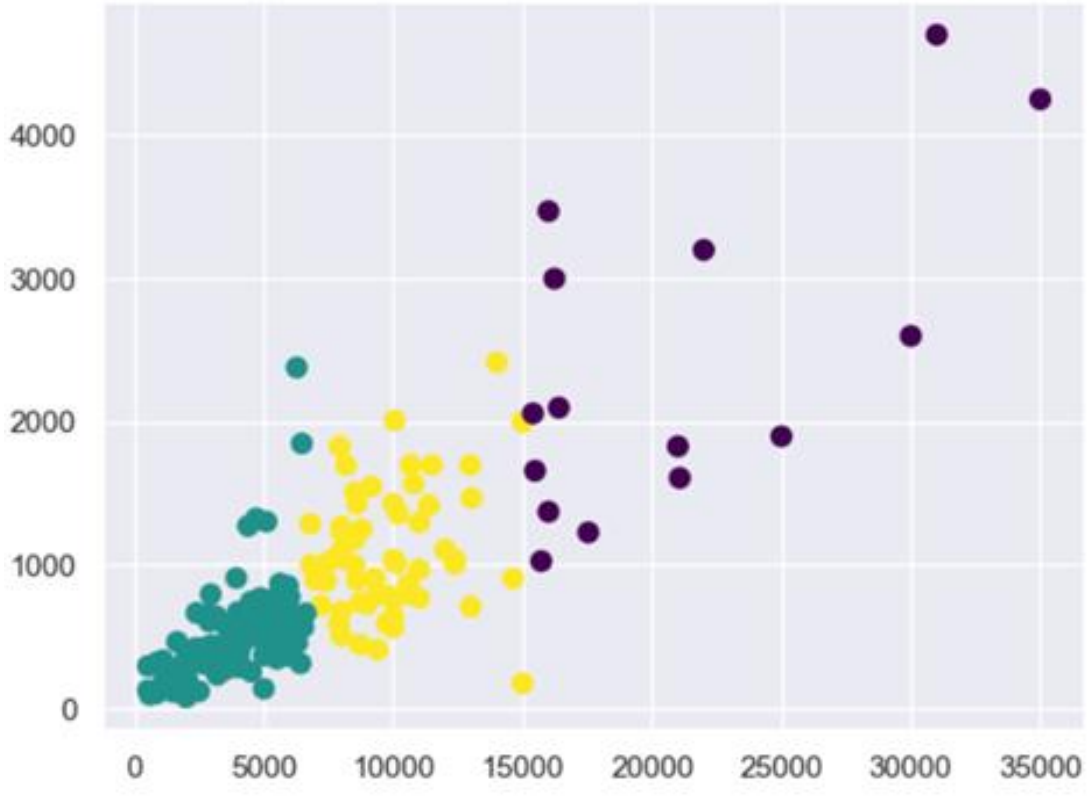



Рисунок 1.23 – Результат кластерного аналізу для трьох кластерів

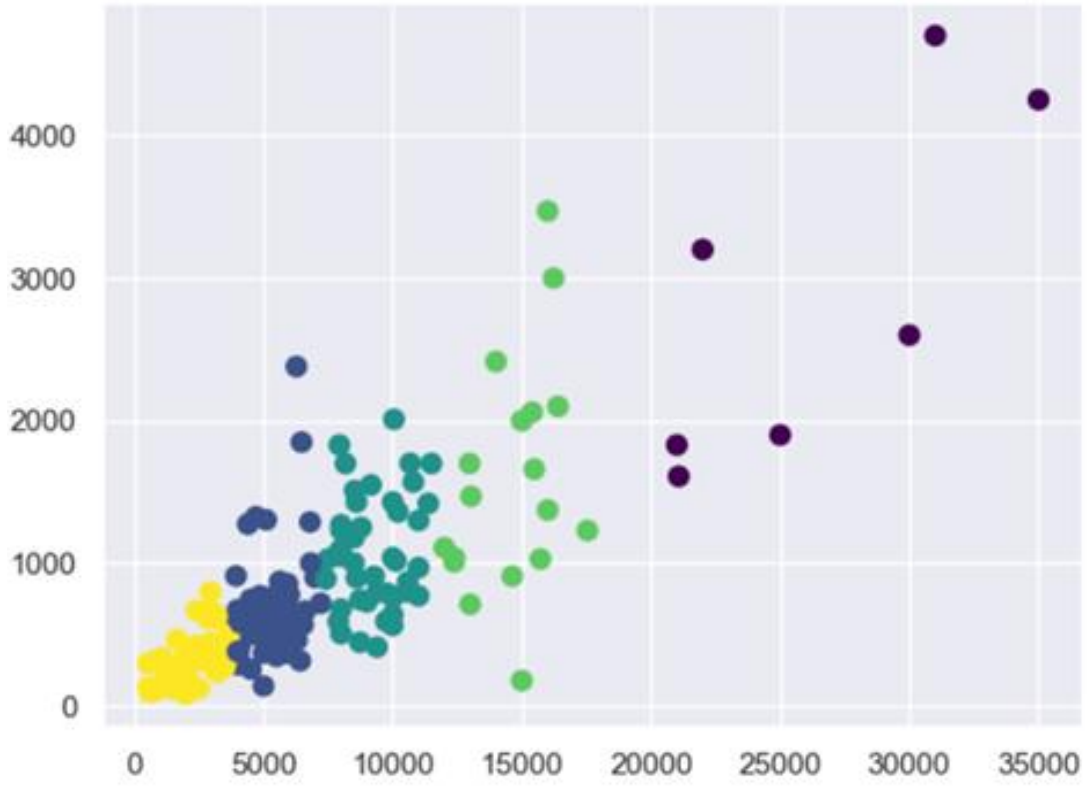



Рисунок 1.24 – Результат кластерного аналізу для п’яти кластерів

### 1.5.3 Завдання для індивідуального виконання роботи

Сформувати множину числових векторів, що визначають описи об’єктів. Реалізувати алгоритм групування [9]:

1) за нормою векторів з використанням медіани або інших статистичних величин;

2) на основі просіювання (алгоритм Уішарта);

3) ієрархічна класифікація на основі міри близькості $\rho(G_1, G_2)$ між кластерами $G_1, G_2$ та заданим порогом близькості;

4) за методом к-середніх [12].

Порівняти результати кластеризації різними методами. Застосувати статистичний пакет для здійснення кластеризації.

### 1.5.4 Контрольні запитання та завдання

1. Пояснити поняття «кластеризація».
2. Пояснити різницю між «мірою близькості» та «метрикою» стосовно кластеризації.
3. Пояснити, як і де застосовується міра близькості між кластерами.
4. Пояснити для чого в ієрархічній класифікації потрібен поріг.
5. Пояснити суть агломеративного та дивізимного способів кластеризації.
6. Пояснити у чому суть алгоритма Уішарта.
7. Чим можна пояснити різні результати кластеризації для однакових вхідних даних?

### 1.6 Комплекс завдань з основ інтелектуального аналізу даних

Сформувати програмно або за допомогою пакету вибірку із суміші трьох випадкових величин з нормальним розподілом та параметрами $N = \{\theta_k, \sigma_k\}_{k=1}^{3}$, де $\theta_k, \sigma_k$ – задані апріорно математичні очікування та середньоквадратичні відхилення компонентів суміші. Задати та обмежити фіксований діапазон вимірювань.

1. Побудувати гістограму значень з метою візуального аналізу даних.

2. Обчислити середнє значення та середньоквадратичне відхилення вибірки, застосувати критерій $\chi^2$ для перевірки гіпотези про нормальний закон розподілу. Виконати обчислення та аналіз при різних значеннях $\theta_k, \sigma_k$, включаючи варіант однакових значень параметрів суміші.

3. Визначити параметри кореляції та регресії суміші та її окремих компонент.

4. Сформувати дерево рішення для задачі віднесення заданого значення до однієї з компонентів суміші.

5. Розробити набір асоціативних правил для задачі віднесення вимірюваного значення до однієї з компонентів суміші.

6. Застосувати кластерний аналіз двома різними методами для розділення елементів на компоненти суміші із заданою або невизначеною кількістю таксонів.

7. Виконати оцінку значень порогів для розбивання суміші та обчислити показники якості розбивань у вигляді кореляційного відношення при аналізі даних.

8. Застосувати пакет *STATISTICA* для реалізації задач ІАД.

9. Розробити програмний проєкт для формування списку здобувачів вищої освіти освітньо-кваліфікаційного рівня «магістр» (бюджет та контракт) на основі таких факторів: кількість студентів, кількість бюджетних місць, правило формування (сума балів по рейтингу, за захист кваліфікаційної роботи здобувача вищої освіти освітньо-кваліфікаційного рівня «бакалавр», з екзамену по іноземній мові та фахового екзамену, бал за науково-дослідну роботу).

10. Розробити програмний проєкт для формування списку студентів трьох спеціальностей (бюджет та контракт) на основі факторів: кількість абітурієнтів, кількість бюджетних місць, правило формування (сума балів атестату, трьох тестів, вибору рейтингу спеціальності (1, 2, 3), права на першочерговий вступ).

# 2 ВИРІШЕННЯ ІНТЕЛЕКТУАЛЬНИХ ЗАДАЧ

## 2.1 Розпізнавання візуальних об’єктів у системах комп’ютерного зору

*Розпізнавання образів* – процес віднесення вихідних даних (образу візуального об’єкта) до певного класу на підставі обумовленого правила [16].

*Образ* – модель, яка відбиває властивості розпізнаваного об’єкта.

*Ознака* – характеристика об’єкта [17].

*Система розпізнавання образів* – комп’ютерний комплекс, що реалізує прийняття рішень.

*Вирішальне правило* (класифікатор) – математичний вираз або алгоритм для визначення належності образу до одного із класів.

### 2.1.1 Теоретичні засади

Сприйняття та розпізнавання образів – одна з ключових проблем теорії штучного інтелекту. Сучасні системи комп’ютерного зору моделюють інтелектуальну діяльність людини щодо аналізу зорової інформації. Аналіз включає в себе вирішення ряду задач: сприйняття зображення, виявлення та обчислення ознак об’єктів, розпізнавання об’єктів на зображенні та ряд інших.

На вході до системи комп’ютерного зору маємо деяке зображення, на виході – результат у вигляді координат чи вектору ознак об’єктів, назви класу об’єкта та інше [11].

Розглянемо найпростішу систему комп’ютерного зору, що виявляє «куточки» на бінарному зображенні. За кількістю кутів можна, наприклад, прийняти рішення, який геометричний об’єкт містить зображення: квадрат, трикутник, трапеція, довільний багатокутник.

Для виявлення кутів на зображенні розроблено та досліджується різноманіття методів [11, 13]. Пропонується дослідити один із відомих методів, а також спробувати його удосконалити чи створити та реалізувати власний метод. До найбільш досконалих з відомих способів відносять метод SURF [17], який працює з напівтоновими зображеннями та формує у заданій точці вектор інваріантних до геометричних перетворень характеристик. При цьому кожний метод вирішує задачу по-своєму, в той же час, маючи деякі обмеження при застосуванні. Для того, щоб відрізняти прямокутник чи квадрат від трапеції, треба визначити не тільки наявність кутів, але й оцінювати їх величину.

Основна ідея більшості підходів – аналіз локального околу у вигляді матриці $n \times n$ точок зображення. Як правило, параметр $n$ вибирають непарним. При цьому повне зображення включає $N \times N$ пікселів.

Матриця околу сканує зображення, зміщуючись на один стовпець, кожного разу обчислює значення скалярного добутку на підматрицю зображення та приймає рішення, чи знаходиться у цьому положенні зображення кута.

Як правило, в цей момент приймається інтелектуальне рішення на основі деякого апріорно заданого порогу. Вихідна матриця при цьому буде містити $(N-n+1)\times(N-n+1)$ пікселів чи значень відгуків побудованого локального фільтру.

На відміну від детекторів Моравець і Харріса, детектор SUSAN аналізує не градієнт зображення, а яскравість, завдяки чому забезпечується швидкодія алгоритму, завдяки чому його можна використовувати у реальному часі.

Основна ідея SUSAN у тому, що сусіди кожної точки в однорідній області мають близьку до неї яскравість, а поблизу кутів число сусідів з однаковою яскравістю зменшується.

Цей принцип продемонстровано на рисунку 2.1 [18].

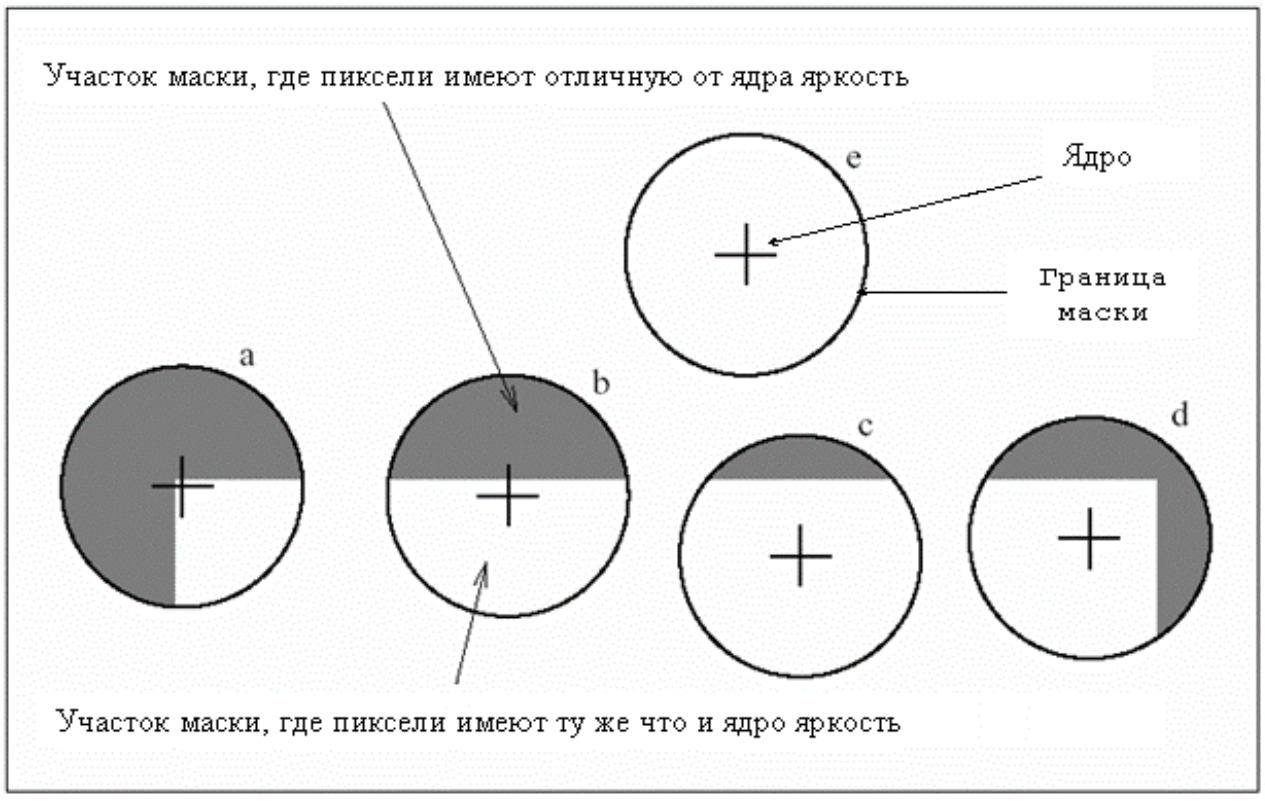



Рисунок 2.1 – Маски, що накладені на зображення

Навколо кожного пікселя зображення будується маска, центральний піксель якої називається ядром (як правило, використовується кругла маска, що включає 37 пікселів). Пікселі в межах маски, які мають порівнянну з ядром яскравість, утворюють область USAN (Univalue Segment Assimilating Nucleus – однорідний сегмент, що асоціюється з ядром). Площа USAN максимальна, коли ядро знаходиться в однорідній (або майже однорідній) області зображення, вона зменшується до половини цього максимуму поблизу лінії кордону і зменшується ще більше поблизу кута, і досягає локальних мінімумів в кутах. Це властивість площі USAN використовується як головний критерій присутності двовимірних особливостей на зображенні.

Маска поміщається в кожну точку зображення, і яскравість кожної точки маски порівнюється з ядром (центральною точкою). Таке порівняння має вигляд

$$c(r,r_0)=\begin{cases}1, if\ \left|I(r)-I(r_0)\right|\le t,\\ 0, if\ \left|I(r)-I(r_0)\right|>t,\end{cases} \tag{2.1}$$

де $r_0$ – позиція ядра маски на зображенні;

$r$ – положення інших точок;

$I(r)$ – яскравість зображення в точці $r$;

$t$ – поріг яскравості.

Порівняння робиться для кожної точки маски, в результаті підраховуємо

$$n(r_0)=\sum_r c(r,r_0). \tag{2.2}$$

Таким чином, $n$ дає кількість пікселів (площа) в USAN. Параметр $t$ різний для кожного зображення і повинен бути налаштованим.

На першому етапі оброблення отримуємо нову матрицю менших розмірів, ніж початкове зображення. Для завершення процедури детектування точок інтересу ця матриця оброблюється, щоб вибрати в ній мінімальні відгуки із застосуванням «немаксимального придушення».

*Немаксимальне придушення* – це додаткова спеціальна логічна обробка отриманих відгуків – евристика, яка у заданому вікні (наприклад, $7\times 7$) залишає тільки значущі мінімуми. Це мінімуми, що знаходяться в центрі вікна, тобто в околі яких всередині маски решта значень більша за них.

Таким чином, процедура детектування точок інтересу виконується у два етапи. Спочатку шляхом сканування зображення маскою обчислюється матриця значень (2.2), а далі ця матриця сканується іншою маскою з метою визначення значущих мінімумів. Їх координати вважаються точками інтересу.

Для стабільних і правильних результатів, особливо при обробці градієнтних зображень, замість функції (1) доцільніше використовувати

$$c(r, r_0) = \exp\left( -\left( \frac{I(r) - I(r_0)}{t} \right)^6 \right). \qquad (2.3)$$

### 2.1.2 Особливості розпізнавання візуальних об’єктів

Розглянемо приклади роботи детекторів точок інтересу для зображення, що не містить складного фону (рис. 2.2, а)).

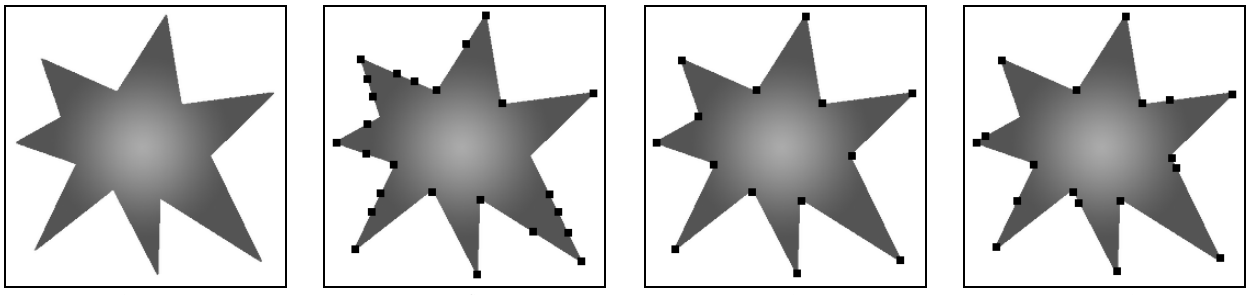
На рисунку 2.2, б) наведено результат роботи детектора Моравець, який виділив 25 точок інтересу.

Як бачимо, детектор Моравець помилково спрацьовує на лініях, що мають «зубчасту» структуру через дискретизацію.

На рисунку 2.2, в) наведено результат роботи детектора Харріса, який виділив 14 точок інтересу.

На рисунку 2.2, г) наведено результат роботи детектора SUSAN, який виділив 18 точок інтересу. Тут прийнято $t = 10$.

Зауважимо, що для всіх детекторів виконувалося немаксимальне придушення маскою $7 \times 7$.



а) б) в) г)

Рисунок 2.2 – Дія детекторів точок інтересу:
а) зображення; б) детектор Моравець; в) детектор Харріса; г) детектор SUSAN

### 2.1.3 Завдання для індивідуального виконання роботи

Робота включає такі дії: введення зображення у пам’ять комп’ютера, виведення аналізованого зображення на екран, реалізація алгоритму аналізу зображення, виведення результату у вигляді нового зображення або повідомлень про місцезнаходження кутів як особливих точок.

Необхідно на практиці реалізувати розпізнавання геометричних фігур: квадрат, прямокутник, правильний та прямокутний трикутник, ромб, трапеція, довільні п’яти- та шестикутник.

### 2.1.4 Контрольні запитання та завдання

1. У чому полягає суть інтелектуальності системи комп’ютерного зору?
2. Пояснити поняття «точка інтересу зображення» та навести приклади практичних застосувань.
3. Пояснити суть детектора SUSAN для формування точок інтересу зображення.
4. Пояснити чому різні детектори виділяють різні множини точок інтересу.
5. Як можна використовувати точки інтересу для розпізнавання геометричних фігур?
6. Пояснити у чому суть взаємодії зображення та маски для визначення точки інтересу.
7. Пояснити суть процедури немаксимального придушення.

## 2.2 Пошук рішень інтелектуальних задач

Вирішення задач є фундаментальним для більшості застосунків штучного інтелекту. Здатність вирішувати задачі часто використовується як міра інтелектуальності як для людей, так і для комп'ютерних систем.

Існує два основних типи завдань [6].

Завдання першого типу можуть бути вирішені з використанням детермінованої процедури, яка буде гарантовано успішною. Для завдань цього типу існують традиційні алгоритми рішення.

Більш значний клас завдань становить другу категорію, тобто завдання, які можуть бути вирішені шляхом пошуку рішення. Цей підхід до вирішення завдань є предметом штучного інтелекту.

### 2.2.1 Теоретичні засади

Одна з найбільш складних перешкод, яке необхідно подолати при спробах застосування методів штучного інтелекту до реальних завдань – це величезна розмірність і складність більшості практичних ситуацій. Завжди є необхідність у розробленні результативних методів пошуку, тому що вважається, що пошук є основним елементом рішення задачі і він є найбільш важливим компонентом інтелекту.

Найбільш застосовні в штучному інтелекті такі методи пошуку: *повний перебір, пошук в глибину, пошук в ширину, пошук з підйомом на пагорб, пошук за найменшою вартістю* [5, 7, 16].

При оцінюванні методів пошуку слід звернути увагу на два ключових питання:

1) як швидко метод призводить до вирішення задачі;

2) наскільки ефективним є отримане рішення.

Моделювання та вивчення особливостей методів будемо здійснювати на прикладі задачі покупки квитка з Нью-Йорку до Лос-Анджелеса у відповідності до заданого списку рейсів:

Нью-Йорк – Чикаго – 1000 миль
Чикаго – Денвер – 1000 миль
Нью-Йорк – Торонто – 800 миль
Нью-Йорк – Денвер – 1900 миль
Торонто – Калгарі – 1500 миль
Торонто – Лос-Анджелес – 1800 миль

Торонто – Чикаго – 500 миль
Денвер – Урбана – 1000 миль
Денвер – Хьюстон – 1500 миль
Хьюстон – Лос-Анджелес –1500 миль
Денвер – Лос-Анджелес – 1000 миль

Цей список рейсів можна подати як направлений граф, де стрілки вказують напрямок руху (рис. 2.3).


Рисунок 2.3 – Направлений граф для списку рейсів

Граф можна представити також у вигляді дерева, де міста з’являються більше одного разу, щоб спростити побудову маршрутів та подальший аналіз (рис. 2.4).

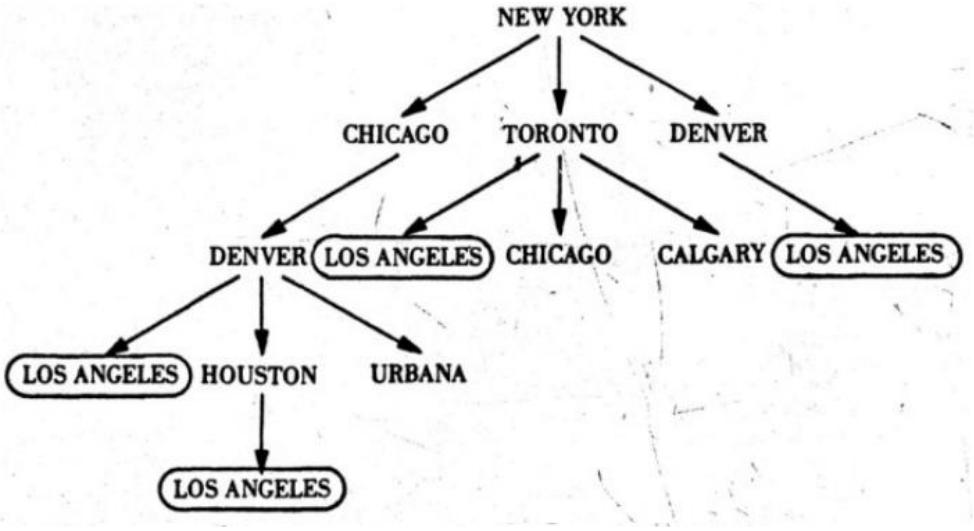



Рисунок 2.4 – Дерево даних для вирішення задачі

*Пошук в глибину* досліджує кожний можливий шлях до самого кінця, перш ніж перейде до іншого шляху, якщо мета не була досягнута. При цьому типі пошуку спочатку з кожної вершини вибирається крайня ліва дуга. Просування здійснюється до тих пір, поки не буде досягнуто або термінальний вузол або мета. Якщо зустрівся термінальний вузол, то необхідно повернутися на один рівень назад і повернути праворуч, а потім продовжувати триматися лівої сторони, поки не зустрінеться або мета, або новий термінальний вузол.

Якщо для якогось проміжного вузла всі вихідні дуги досліджені, і вони ведуть в тупик, то повертаємося до попереднього вузла. Процес повторюється до тих пір, поки не буде знайдена мета чи будуть пройдені всі вузли з області пошуку. Пошук в глибину гарантовано знаходить мету (якщо вона існує), тому що в найгіршому випадку він переходить у повний перебір.

При пошуку в глибину отримано такий результат: New York – Chicago – Denver – Los Angeles, distance is 3000. У той же час оптимальне рішення є: Нью-Йорк, Торонто, Лос-Анджелес – відстань 2600 миль.

У розглянутому прикладі пошуком в глибину знайдено досить гарне рішення при першій же спробі і без будь-якого повернення. Однак, для досягнення оптимального рішення необхідно було б пройти майже всі вузли, що зовсім добре. Пошук в глибину може слабо спрацювати в ситуаціях, коли необхідно досліджувати особливо завдання, що не має рішення. Тут пошук в глибину буде витрачати даремно чимало часу.

*Пошук в ширину* працює з етапами руху (рис. 2.4) та перевіряє кожен вузол на одному рівні, перш ніж переходить до наступного рівня. Пошук в ширину гарантує виявлення рішення, якщо воно існує, тому що в кінцевому рахунку вироджується в повний перебір.

Метод пошуку в ширину дає наступне рішення: New York – Toronto – Los Angeles, distance is 2600. Воно є оптимальним рішенням.

Однак, це не можна узагальнювати на інші ситуації, тому що результат залежить від організації інформації. Цей результат ілюструє радикальну відмінність пошуків в глибину і в ширину. Недоліки пошуку в ширину проявляються, коли мета розташована на кілька рівнів глибше. У цьому випадку пошук в ширину докладає значне зусилля. Програміст вибирає між пошуком в глибину і в ширину, роблячи припущення щодо того, де найбільш імовірно знаходиться ціль.

У задачі планування польоту з Нью-Йорка в Лос-Анджелес є два обмеження, які пасажир захоче мінімізувати.

Перше – число пересадок.

Друге – довжина маршруту.

Причому найкоротший маршрут не обов'язково має найменшу кількість пересадок. Алгоритм пошуку, що мінімізує число пересадок, використовує *евристику*, яка вважає, що чим більша відстань покривається, тим вище імовірність того, що Ви переміститеся ближче до місця призначення і, таким чином, скорочується число пересадок. У штучному інтелекті цей метод пошуку називається «*сходження на пагорб*».

Для завдання про авіарейси можна включити евристику для алгоритму «сходження на пагорб»: виберіть суміжний рейс, місце призначення якого є якнайдалі від поточного положення, в надії, що це буде ближче до кінцевої мети.

Для контрольного прикладу отримаємо рішення: New York to Denver to Los Angeles, distance is 2900.

Стосовно до задачі про авіарейси пошук за мінімальною вартістю передбачає, що програма буде брати найкоротший рейс у всіх випадках так, щоб знайдений маршрут мав хороший шанс бути найкоротшим. На відміну від пошуку «сходженням на пагорб», який мінімізує число пересадок, пошук за мінімальною вартістю мінімізує протяжність маршруту.

Рішення: New York to Toronto to Los Angeles, distance is 2600.

Тут пошук за мінімальною вартістю знаходить найкоротший маршрут. Пошук за мінімальною вартістю має ті ж самі переваги і недоліки, що і «сходження на пагорб». Можуть зустрічатися помилкові долини, низини і ущелини, але в цілому пошук за мінімальною вартістю має тенденцію працювати ефективно.

Якщо необхідно отримати майже оптимальне рішення, але немає можливості застосувати вичерпний пошук (повний перебір), то слід спробувати кожну з технологій пошуку і далі використовувати найбільш підходящу. Це правило найкраще, тому що робота різних видів пошуку істотно різна на різних завданнях.

Термін «оптимальне рішення» зазвичай використовується для позначення найкращого маршруту, який Ви можете знайти, використовуючи одну з різних технологій знаходження множинних рішень. Фактично воно може і не бути найкращим рішенням.

Виявлення дійсно оптимального рішення вимагало б гранично дорогого вичерпного пошуку.

### 2.2.2 Завдання для індивідуального виконання роботи

Запропонована технологія включає такі дії: введення даних про маршрути та їх вартості, побудова графа та дерева маршрутів, моделювання одного із методів пошуку: повний перебір, пошук в глибину, пошук в ширину, пошук з підйомом на пагорб, пошук за найменшою вартістю, оптимальний пошук, множинний пошук. Виконати порівняння результатів оброблення для різних варіантів даних. Зробити висновки про результативність методів пошуку рішень у системах штучного інтелекту.

### 2.2.3 Контрольні запитання та завдання

1. Пояснити у чому полягає інтелектуальність методів пошуку рішень задач.
2. Що таке евристика?
3. Пояснити суть методів: пошук в глибину, пошук в ширину.
4. Пояснити суть методів: пошук з підйомом на пагорб, пошук за найменшою вартістю.
5. Пояснити, яким чином можна вибрати найкращий із методів пошуку рішень задач.
6. Пояснити поняття «оптимальне рішення».

## 2.3 Прийняття рішень в експертних системах

Експертна система – інтелектуальна комп'ютерна програма, що використовує знання та аналітичні здібності кількох експертів (людей) щодо деякої галузі застосування, і здатна робити логічні висновки на основі цих знань, забезпечуючи вирішення прикладних завдань [5].

### 2.3.1 Теоретичні засади

Важливим напрямком при оцінюванні, наприклад, собівартості розробленого програмного забезпечення або при прийнятті персоналу на вакантну посаду є застосування методів експертного оцінювання, так як цей підхід є базовим при відсутності повного формалізованого подання аналізованих даних.

Суть методів експертного оцінювання полягає в організації роботи зі спеціалістами-експертами і аналітичному обробленні сукупності думок експертів.

Питання, пов'язані з експертизою, розглядаються і вирішуються дослідником (консультантом). Він визначає множину допустимих оцінок $\Omega$, множину $\Omega_e$ допустимих оцінок експертів (із якої реалізують вибір експерти), параметри схеми експертизи, виконує підбір експертів, організовує реалізацію експертизи та ін.

Існує ряд методів побудови системи експертного оцінювання. Аналіз експертиз показує, що у процесі їх створення можна виділити таку основну послідовність дій [7].

1. Визначаються множини допустимих оцінок $\Omega$, $\Omega_e$.

2. Кожний експерт вибирає свою оцінку $a_i = C_i(\Omega_e) \in \Omega_e$, $i = \overline{1, N}$, тобто вирішує задачу вибору найкращої оцінки із $\Omega_e$. При цьому експерти іноді можуть взаємодіяти між собою, хоча більшість практичних підходів це виключають.

3. Відповідно до раніше розробленої процедури (формули) дослідник виконує оброблення $\varphi(C_1(\Omega_e), C_2(\Omega_e), ..., C_N(\Omega_e))$ отриманої від експертів інформації і знаходить результуючу оцінку із $\Omega$, тобто виконує відображення $\varphi : \Omega_e \to \Omega$, що є фактичним вирішенням задачі оцінювання.

4. Якщо отримане рішення не задовольняє дослідника, він може представити експертам додаткову інформацію, тобто організувати зворотній зв'язок, після чого вони знову вирішують задачу вибору.

Зважаючи на те, що людина-експерт значно легше сприймає та аналізує операцію порівняння даних, ніж точне визначення числових оціночних значень, у методах експертного оцінювання отримало поширення рангове оброблення даних, що зводиться до розміщення об'єктів у порядку зменшення їх якості, що відповідає деякому критерію. Якість об'єкта відповідає номеру у перестановці номерів об'єктів, причому об'єкт з номером «1» вважається найкращим. Результати опитування експертів зводяться в таблицю (табл. 2.1).

Таблиця 2.1 – Результати опитування експертів

| Експерти | Об'єкти | | | |
|---|---|---|---|---|
| 1 | $r_{11}$ | $r_{12}$ | … | $r_{1n}$ |
| 2 | $r_{21}$ | $r_{22}$ | … | $r_{2n}$ |
| … | | | | |
| $N$ | $r_{N1}$ | $r_{N2}$ | … | $r_{Nn}$ |
| Сума рангів | $r_1$ | $r_2$ | … | $r_n$ |

Розглядають два варіанти побудови можливих порівняльних оцінок експертів: строге та нестроге ранжирування. Строге ранжирування означає, що у кожного експерта всі об'єкти отримують різні ранги.

У цьому випадку $\Omega = \Omega_e$ і включає множину перестановок із $n$ об'єктів при умові ізольованості експертів та відсутності зв'язку між ними.

У рядку таблиці 2.1 з номером $i$ стоять місця (ранги), проставлені $i$-м експертом об'єктам, що ним упорядковуються. У $(N+1)$-му рядку стоять обчислені суми рангів, отриманих об'єктами від усієї множини експертів.

Далі усі $n$ об'єктів упорядковуються у відповідності до значення $r_q$, що визначається за формулою

$$r_q = \sum_{j=1}^{N} r_{qj} . \qquad (2.4)$$

На перше місце як найкращий ставиться об'єкт, у якого $r_q$ мінімальне.

Ступінь узгодженості експертів (для випадку строгого ранжирування) визначається за допомогою коефіцієнта конкордації $w$.

$$W = \frac{12\sum_{i=1}^{n}[r_i - \frac{1}{2}N(n+1)]^2}{N^2(n^3-n)}, \qquad (2.5)$$

де $n$ – число об'єктів;

$N$ – число експертів.

Коефіцієнт конкордації змінюється в діапазоні $0 < W < 1$, причому «0» відповідає ситуації повної неузгодженості експертів, а «1» – повній узгодженості їх рішень.

Експериментально встановлено, що значні труднощі у експерта викликають завдання ранжирування на основі одночасного врахування декількох різних ознак об'єкта (наприклад, таких різнотипних характеристик, як об'єм бензобака, максимальна швидкість, вартість для автомобіля), за якими оцінюються об'єкти.

У цих випадках переходять до *попарного порівняння*, як правило, не строго ранжируваних об'єктів. У методі парних порівнянь кожний із експертів виконує $C_n^2$ порівнянь усіх можливих пар об'єктів.

Результат порівняння $j$-го експерта подається матрицею розміром $n \times n$, у якій елемент $a_{ik} = 1$, тоді і тільки тоді, коли на думку експерта $i$-й об'єкт має перевагу над об'єктом $k$.

Для будь-якої пари об'єктів один з них обов'язково має перевагу, а $a_{ii} = 0$ (діагональні елементи) за визначенням.

Матриця $A^j$, представлена $j$-м експертом, є матрицею деякого бінарного відношення, що називається відношенням переваг експерта.

Відображення $\varphi$ для цього випадку задається наступним чином. Обчислюють матрицю $A = \sum_{j=1}^{N} A^j$, далі визначають величини $a_s = \sum_{i=1}^{n} a_{is}$, $s = \overline{1,n}$, об'єкти упорядковують у відповідності з величинами $a_s$, тобто об'єкт з мінімальним $a_s$ отримує ранг «1» і т.д.

### 2.3.2 Приклади обчислення та візуалізації значень експертних оцінок

Розглянемо застосування методу *мінімальної суми рангів* на прикладі уявних експертних даних про якість цукерок, вироблених представленими виробниками (табл. 2.2).

Таблиця 2.2 – Вихідні дані експертів

| **Експерти** | **Об`єкти-виробники** | | | | |
|---|---|---|---|---|---|
| | Рошен | Світоч | «Красный октябрь» | АВК | Конті |
| 1 | 2 | 3 | 1 | 4 | 5 |
| 2 | 1 | 2 | 4 | 3 | 5 |
| 3 | 4 | 5 | 3 | 2 | 1 |
| 4 | 1 | 2 | 3 | 4 | 5 |
| 5 | 3 | 5 | 4 | 1 | 2 |

Далі обчислюємо у стовпці суму рангів для всієї множини експертів і отримаємо результат, застосовуючи процедуру ранжирування сумарних рангів. На першому місці повинен бути об’єкт, що має мінімальну суму (табл. 2.3).

Таблиця 2.3 – Результат оброблення експертних оцінок

| **Сума рангів** | 11 | 17 | 15 | 14 | 18 |
|---|---|---|---|---|---|
| **Результат** | 1 | 4 | 3 | 2 | 5 |

Щоб дізнатись ступінь узгодженості думок експертів, обчислимо коефіцієнт конкордації (2.5). Отриманий результат покаже рівень узгодженості, що відображає близькість думок експертів між собою.

Тепер розглянемо *метод попарного порівняння*.

На основі даних таблиці 2.2 будуємо матрицю $A^j$ для кожного експерта. Для цього попарно порівнюємо кожен об’єкт експертизи.

Результат порівняння 1-го експерта подається матрицею розміром 5×5, у якій елемент $a_{ik} = 1$, тоді і тільки тоді, коли на думку експерта $i$-й об’єкт має перевагу над об’єктом $k$.

В інших випадках ставимо значення «0» (табл. 2.4).

Таблиця 2.4 – Матриця порівняння для 1-го експерта

| | | | | | |
|---|---|---|---|---|---|
| $A_1 =$ | 0 | 1 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 1 | 1 |
| | 1 | 1 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 0 | 1 |
| | 0 | 0 | 0 | 0 | 0 |

У такому ж порядку будуємо матрицю для кожного експерта.

Далі виконуємо додавання 5-ти отриманих матриць.

У сумарній матриці $A$ обчислюємо суму по стовпцям (табл. 2.6).

Таблиця 2.5 – Матриця $A$

| | | | | | |
|---|---|---|---|---|---|
| $A=$ | 0 | 5 | 3 | 3 | 3 |
| | 0 | 0 | 2 | 3 | 3 |
| | 2 | 3 | 0 | 2 | 3 |
| | 2 | 2 | 3 | 0 | 4 |
| | 2 | 2 | 2 | 1 | 0 |
| **Сума** | 6 | 12 | 10 | 9 | 13 |

Далі обчислюємо ранг кожного об’єкта. Значенню з мінімальною сумою по стовпцю присвоюється ранг «1» (табл. 2.6).

Таблиця 2.6 – Обчислення рангу

| | | | | | |
|---|---|---|---|---|---|
| **Сума** | 6 | 12 | 10 | 9 | 13 |
| **Ранг** | 1 | 4 | 3 | 2 | 5 |

Порівнюючи результати експертного оцінювання кожним із розглянутих методів, можна зробити висновок, що результат співпадає. Для загального випадку це не є обов’язковим.

### 2.3.3 Завдання для індивідуального виконання роботи

1. Ознайомитися з роботою наведених прикладів.
2. Встановити та вибрати для прикладної задачі (за вибором студента та узгодження із викладачем) дані, на основі яких буде розрахована рейтингова оцінка.
3. Внести зміни до проєкту у відповідності до завдання. Сформувати та обчислити рейтингову оцінку з використанням наведених методів.
4. Порівняти результати роботи для різних методів.

### 2.3.4 Контрольні запитання та завдання

1. Пояснити у чому полягає інтелектуальність методів побудови експертних оцінок.
2. Пояснити принципову відмінність розглянутих методів.
3. Пояснити суть вимог до експертів, що формують оцінки.
4. Пояснити призначення та застосування експертних оцінок.
5. Чим принципово експертні системи відрізняються від решти інтелектуальних систем?
6. Пояснити поняття «оптимальне рішення».

## 2.4 Моделювання дій з нечіткими числами та множинами

*Нечіткою множиною* *A*, що задана на універсальній множині *E*, називають сукупність пар чисел $(x, \mu_A(x))$, де $\mu_A(x) \in [0,1]$, а $x \in E$. При цьому функцію $\mu_A(x)$ називають функцією належності, що відповідає ступеню належності (істинності) елемента *x* до множини *A* [17, 19].

Для універсальної множини *E* виконується $\mu_E(x) = 1$.

Множину *A* називають *опуклою*, якщо кожна підмножина $A_\alpha$ для $\alpha$-перерізу відповідає умові:

$$A_\alpha = \{x \mid \mu_A(x) \geq \alpha\} \tag{2.6}$$

де $\alpha$ – ступінь належності до нечіткої множини *A*.

Множину *A* називають *нормальною*, якщо виконується таке: максимальне значення серед функцій належності для всіх значень *x* дорівнює «1».

*Нечітким числом* називається нечітка множина *A*, функція належності якої є опуклою та нормальною.

### 2.4.1 Теоретичні засади

Необхідно детально ознайомитися з визначенням та характеристиками нечітких множин, діями над інтервалами довіри, поняттями нечіткого числа та нечіткого трикутного числа, функції належності, правилами дій з додавання та віднімання нечітких чисел трикутного виду, включаючи форму з перерізами у вигляді $\alpha$ [17,19].

Послідовність дій при виконанні роботи.

1. Розробити програмну модель для додавання та віднімання нечітких чисел трикутної форми. Мова програмування вибирається за власним бажанням.

2. Результати дій додавання та віднімання подати спочатку у вигляді

$$\underline{A} = (\underline{a}, \hat{a}, \overline{a}), \tag{2.7}$$

де $\underline{a}$ та $\overline{a}$ – відповідно нижня і верхня границі інтервалу, на якому задана нечітка множина (тобто $\mu_{\underline{A}}(\underline{a}) = 0$, $\mu_{\underline{A}}(\overline{a}) = 0$);

$\hat{a}$ – елемент множини $\underline{A}$, для якого функція належності $\mu_{\underline{A}}(\hat{a}) = 1$.

З використанням

$$A_\alpha(+)B_\alpha = [(\alpha\cdot(\hat{a}-\underline{a})+\underline{a})+(\alpha\cdot(\hat{b}-\underline{b})+\underline{b}),$$
$$(\overline{a}-\alpha\cdot(\overline{a}-\hat{a}))+(\overline{b}-\alpha\cdot(\overline{b}-\hat{b}))], \quad (2.8)$$

$$A_\alpha(-)B_\alpha = [(\alpha\cdot(\hat{a}-\underline{a})+\overline{b}-\hat{b})+\underline{a}-\overline{b},\quad \overline{a}-\underline{b}-\alpha\cdot(\overline{a}-\hat{a}+\hat{b}-\underline{b})], \quad (2.9)$$

при значенні $\alpha = 0$.

Потім обчислити значення функцій належності для суми та різниці двох нечітких чисел на основі

$$\mu_{\underline{A}}(x) = \begin{cases} 0, & при \quad x \le \underline{a},\ x \ge \overline{a}, \\ \dfrac{x-\underline{a}}{\hat{a}-\underline{a}}, & при \quad \underline{a} \le x \le \hat{a}, \\ \dfrac{\overline{a}-x}{\overline{a}-\hat{a}}, & при \quad \hat{a} \le x \le \overline{a}. \end{cases} \quad (2.10)$$

Вивести значення діапазону аргументу та функцій належності.

3. Передбачити можливість розрахунку значення функції належності для кожного із нечітких чисел при довільному значенні координати *x*.

4. Перевірити роботу програми для індивідуальних варіантів нечітких чисел (табл. 2.7 [19]).

Таблиця 2.7 – Варіанти завдань

| Варіант | $\underset{\sim}{A}_1$ | $\underset{\sim}{B}_1$ | $\underset{\sim}{A}_2$ | $\underset{\sim}{B}_2$ |
|---|---|---|---|---|
| 1 | [-1, 4, 7] | [4, 7, 8] | [-5, -2, 0] | [-3, 4, 7] |
| 2 | [-3, -2, 3] | [2, 4, 6] | [0, 1, 6] | [-2, -2, 4] |
| 3 | [-5, 3, 8] | [-1, 3, 7] | [-6, 2, 7] | [4, 5, 8] |
| 4 | [7, 8, 9] | [-2, 0, 4] | [3, 3, 5] | [-4, -2, -1] |
| 5 | [2, 6, 8] | [-3, 5, 6] | [-2, 4, 8] | [-7, -4, 0] |
| 6 | [-3, 5, 10] | [2, 4, 7] | [-1, 7, 8] | [0, 2, 6] |
| 7 | [4, 9, 10] | [-6, 2, 5] | [4, 6, 9] | [-8, -5, 3] |
| 8 | [-6, -3, -1] | [3, 5, 7] | [-6, 0, 4] | [4, 7, 9] |
| 9 | [3, 5, 8] | [-2, -1, 0] | [-6, -4, 2] | [-2, 0, 5] |
| 10 | [-3, 7, 9] | [1, 3, 5] | [1, 1, 6] | [-3, -2, 0] |

5. Написати програму для моделювання процесу бункерування суден, де застосовується дія з віднімання нечітких чисел.

Сутність бункерування полягає в наступному.

На початковому етапі на танкері знаходиться визначена кількість палива (чітке число). Замовлення портів мають нечіткий характер, тому, щоб визначити кількість палива на танкері після обслуговування порту, необхідно від чіткого числа (кількості палива на танкері) відняти нечітке число (замовлення порту).

Після цього залишок палива на танкері також є нечітким числом. Танкер може планувати обслуговування портів до того моменту, поки залишок палива на танкері перевищує замовлення порту. Коли ця умова не буде виконуватись, це означатиме виникнення конфліктної ситуації, і планування обслуговування портів необхідно буде припинити.

У загальній кількості виконаних замовлень останнє (на якому виникла конфліктна ситуація) не враховується.

Процес бункерування продемонстровано на рисунку 2.5.

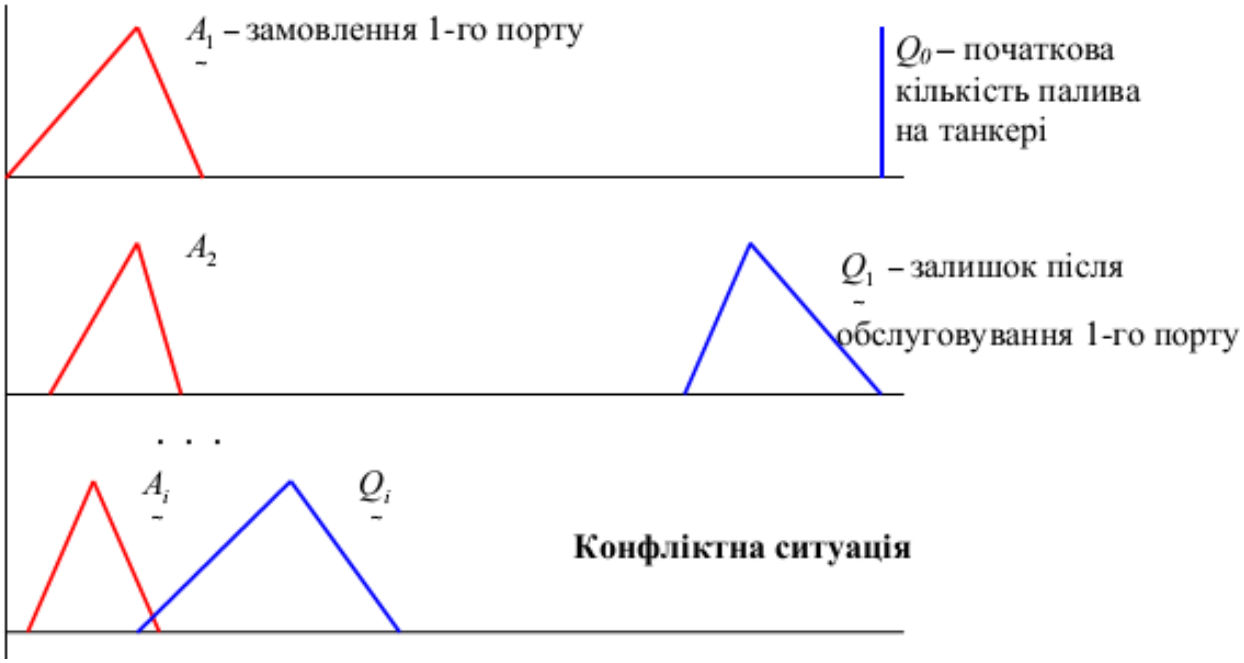



Рисунок 2.5 – Ілюстрація процесу бункерування

Передбачити можливість введення вантажомісткості танкера користувачем. Замовлення портів визначаються з використанням генератора випадкових рівномірних чисел з діапазону 30-50, а нечітке число формується у вигляді інтервалу з відхиленням 5-7, при цьому замовлення порту не перевищує вантажомісткості танкера.

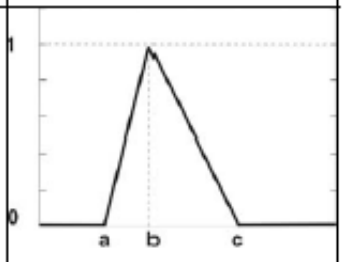
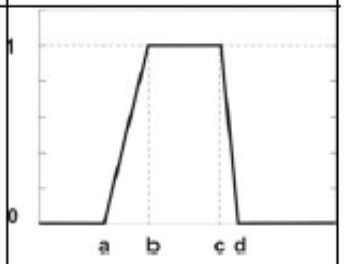
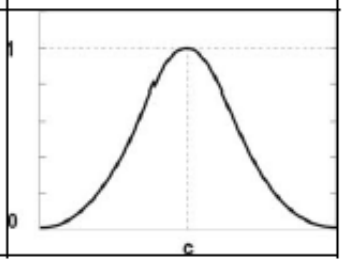
6. Написати програму, в якій змоделювати та графічно показати значення функції належності згідно із варіантом таблиць 2.8 та 2.9 [19].

*Функція належності* – це функція, яка дозволяє обчислити ступінь належності будь-якого елемента до нечіткої множини.

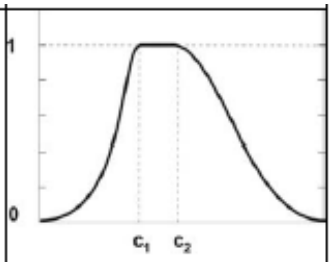
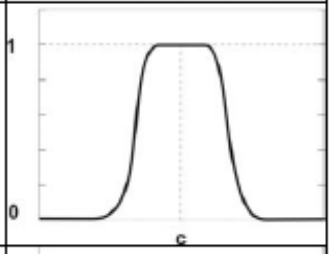
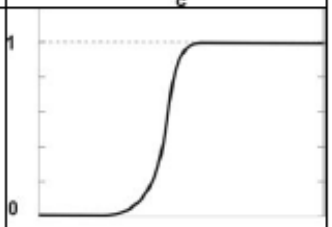
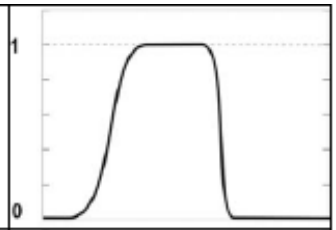
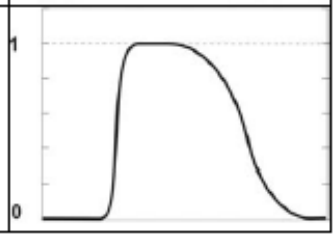
Таблиця 2.8 – Варіанти завдань

| Варіант | Функції належності (№№ із таблиці 2.9) |
|---|---|
| 1 | 1, 4, 5 |
| 2 | 2, 3, 7 |
| 3 | 6, 8, 12 |
| 4 | 1, 3, 13 |
| 5 | 2, 4,12 |
| 6 | 3, 5, 8 |
| 7 | 1, 6, 13 |
| 8 | 4, 5, 7 |
| 9 | 2, 6, 12 |
| 10 | 7, 8, 13 |

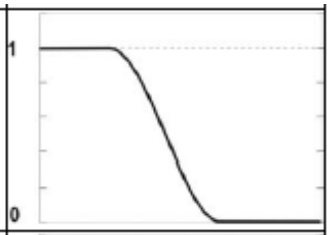
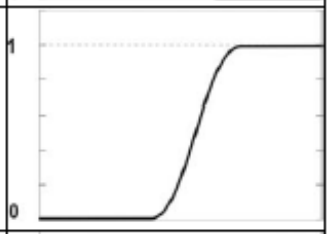
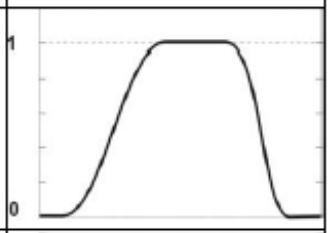
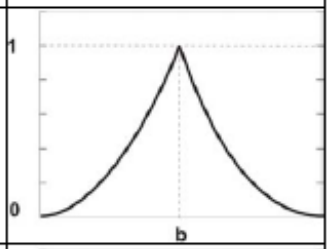
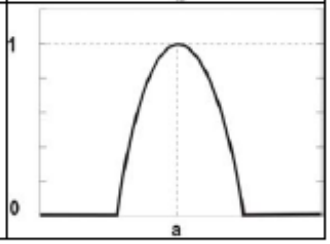
Таблиця 2.9 – Стандартні функції належності нечітких чисел

| № | Опис | Аналітична формула | Вигляд функції належності |
|---|---|---|---|
| 1 | Трикутна функція належності | $\mu(x)=\begin{cases}0, & \text{при } x\le a \text{ або } x\ge c\\ \dfrac{x-a}{b-a}, & \text{при } a\le x\le b\\ \dfrac{c-x}{c-b}, & \text{при } b\le x\le c\end{cases}$, де $a\le b\le c$ |   |
| 2 | Трапеціє-видна функція належності | $\mu(x)=\begin{cases}0, & \text{при } x\le a \text{ або } x\ge d\\ \dfrac{x-a}{b-a}, & \text{при } a\le x\le b\\ 1, & \text{при } b\le x\le c\\ \dfrac{d-x}{d-c}, & \text{при } c\le x\le d\end{cases}$, де $a\le b\le c\le d$ |   |
| 3 | Симетрична гаусівська функція належності | $\mu(x)=e^{\frac{(x-b)^2}{2c^2}}$ |   |

Продовження таблиці 2.9

| № | Опис | Аналітична формула | Вигляд функції належності |
|---|---|---|---|
| 4 | Двобічна гаусівська функція належності | якщо $c_1 < c_2$, то $\mu(x) = \begin{cases} e^{-\frac{(x-c_1)^2}{2a_1^2}}, & \text{при } x < c_1 \\ 1, & \text{при } c_1 \le x \le c_2 \\ e^{-\frac{(x-c_2)^2}{2a_2^2}}, & \text{при } x > c_2 \end{cases}$; якщо $c_1 > c_2$, то $\mu(x) = \begin{cases} e^{-\frac{(x-c_1)^2}{2a_1^2}}, & \text{при } x < c_2 \\ e^{-\frac{(x-c_1)^2}{2a_1^2}} \cdot e^{-\frac{(x-c_2)^2}{2a_2^2}}, & \text{при } c_2 \le x \le c_1 \\ e^{-\frac{(x-c_2)^2}{2a_2^2}}, & \text{при } x > c_1 \end{cases}$ |   |
| 5 | Узагальнена дзвоно-подібна функція належності | $\mu(x) = \frac{1}{1+\left\lvert\frac{x-c}{a}\right\rvert^{2b}}$, де $a \in (0;\ +\infty);\ b \in (-\infty;\ +\infty);\ c \in (-\infty;\ +\infty)$ |   |
| 6 | Сигмоїдна функція належності | $\mu(x) = \frac{1}{1+e^{-a(x-c)}}$ |   |
| 7 | Добуток сигмоїдних функцій належності | $\mu(x) = \frac{1}{1+e^{-a_1(x-c_1)}} \cdot \frac{1}{1+e^{-a_2(x-c_2)}}$ |   |
| 8 | Різниця між сигмоїдними функціями належності | $\mu(x) = \frac{1}{1+e^{-a_1(x-c_1)}} - \frac{1}{1+e^{-a_2(x-c_2)}}$ |   |

Продовження таблиці 2.9

| № | Опис | Аналітична формула | Вигляд функції належності |
|---|---|---|---|
| 9 | Z-подібна функція належності | $\mu(x)=\begin{cases}1, & \text{при } x \le a\\ \text{нелінійна апроксимація}, & \text{при } a<x<b\\ 0, & \text{при } x \ge b\end{cases}$ |  |
| 10 | S-подібна функція належності | $\mu(x)=\begin{cases}0, & \text{при } x \le a\\ \text{нелінійна апроксимація}, & \text{при } a<x<b\\ 1, & \text{при } x \ge b\end{cases}$ |  |
| 11 | π -подібна функція належності | Добуток Z-подібної та S-подібної функцій належності |  |
| 12 | Лапласівська функція належності | $\mu(x)=e^{-\frac{\lvert x-b\rvert}{d}}$, де $d>0$ |  |
| 13 | Квадратична функція належності | $\mu(x)=\begin{cases}1-\left(\frac{x-a}{b}\right)^2, & \text{якщо } \left(\frac{x-a}{b}\right)^2<1\\ 0, & \text{інакше}\end{cases}$ |  |

Передбачити можливість настроювання параметрів функцій належності користувачем.

### 2.4.2 Завдання для індивідуального виконання роботи

Варіанти завдань вибираються з таблиць 2.8 та 2.9.

### 2.4.3 Контрольні запитання та завдання

1. Пояснити основне призначення нечітких множин.
2. Дайте визначення поняттям «нечітка множина», «функція належності».
3. Дайте визначення поняттям «нечітке число», «нечітке трикутне число».
4. Пояснити, яким чином додаються нечіткі числа.
5. Пояснити, що таке «переріз значень» для нечіткого числа.
6. Пояснити у чому нечітка сутність процесу бункерування.
7. Пояснити, яким чином можна обчислити значення функції належності.

## 2.5 Робота з базами знань у Visual Prolog. Обчислення виразів

*База знань* – це спеціалізована база даних, розроблена для управління знаннями (метаданими), тобто збором, зберіганням, пошуком і видачею знань. Розділ штучного інтелекту, що вивчає бази знань і методи роботи зі знаннями, називається *інженерією знань* [20].

Під базою знань розуміється сукупність фактів і правил виведення, що допускають логічні висновки і осмислену обробку інформації. Наприклад, у мові логічного програмування Visual Prolog бази знань описуються у формі конкретних фактів і правил логічного виведення, а також обробляються відповідними процедурами [21].

У відповідях на найпростіші запити до баз знань для системи логічного програмування Visual Prolog відображається значення «істина» або «хибність» залежно від наявності відповідних фактів [22].

Найбільш важливий параметр бази знань – якість знань. Кращі бази знань містять релевантну, достовірну та актуальну інформацію, мають довершені системи пошуку інформації та ретельно продуману структуру і формат знань.

### 2.5.1 Теоретичні засади

В даний час мова логічного програмування Visual Prolog використовується для створення систем управління ресурсами великих комплексів (зокрема, аеропортів), обробки текстів на природній мові, експертних систем, систем медичної діагностики та інших [23].

Мова Visual Prolog є об’єктно-орієнтованою. Програмування в об’єктно-орієнтованому стилі використовується досвідченими програмістами. Тому в основному застосовуються логічний і функціональний стилі програмування, а також консольні застосунки.

Крім цього, використовується інтерпретатор цієї мови PIE, написаний на мові Visual Prolog. Є приклади їх спільного використання.

Початкове меню Visual Prolog 7.5 має вигляд (рис. 2.6). Користувач має можливість відкрити існуючий або створити новий проєкт.

Проєкт створюється так, як показано на рисунку 2.7.

Робота у консольному застосунку ведеться зі змістом файлу `main.pro` (рис. 2.8).

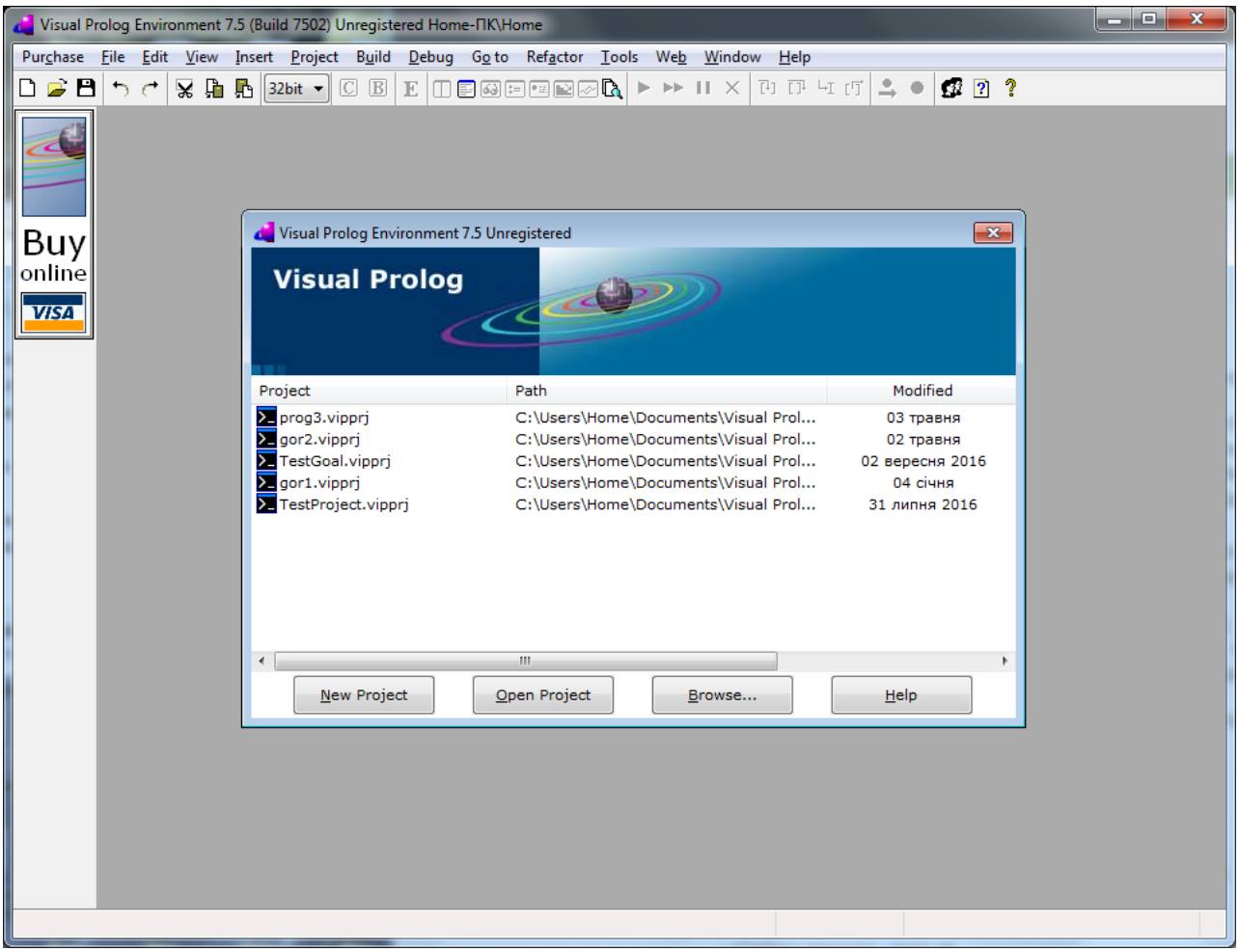



Рисунок 2.6 – Діалогове вікно Visual Prolog 7.5

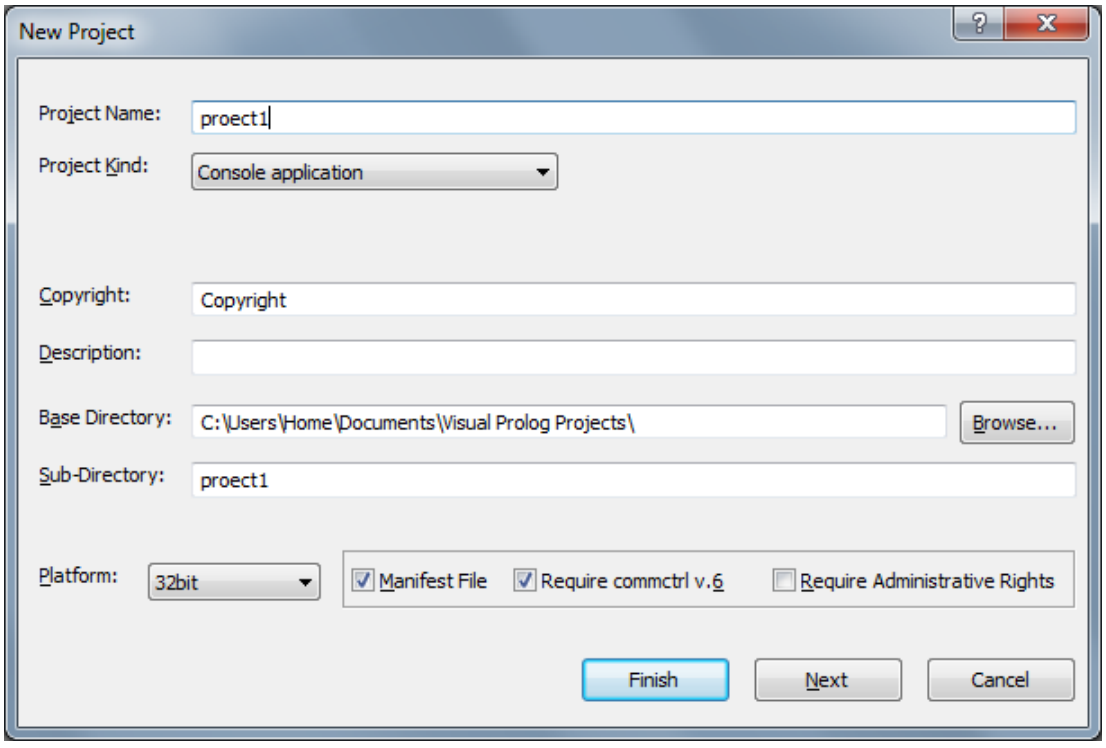



Рисунок 2.7 – Вікно створення проєкту у Visual Prolog 7.5

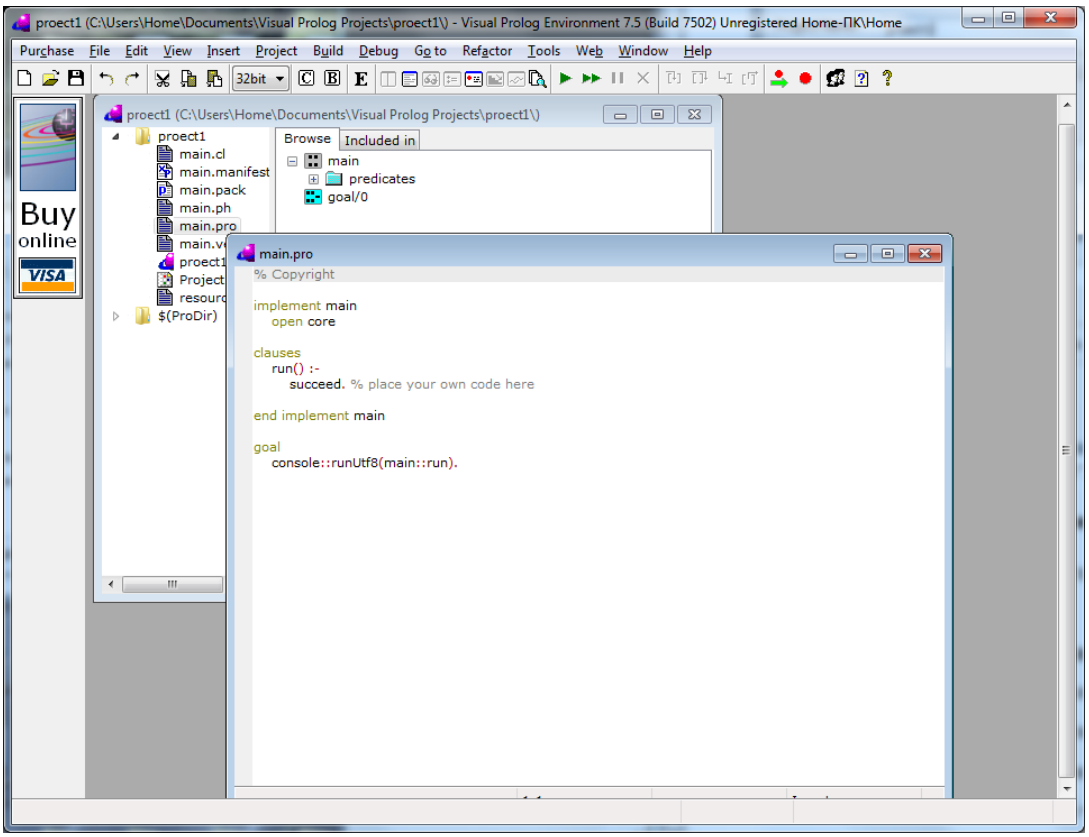



Рисунок 2.8 – Консольний застосунок у Visual Prolog 7.5

Посиланням [21] можна скористуватися для аналізу ряду простих консольних проєктів.

### 2.5.2 Приклади вирішення конкретних задач роботи з базою знань

Розглянемо вирішення конкретної задачі роботи з базою знань «Розклад дня», яка містить ряд фактів з вказівкою відрізку часу та виду дій.

**Завдання № 1.**

*Задано факти:*

заняття(0, 7, сон),
заняття(7, 8, сніданок),
заняття(8, 13, школа),
заняття(13, 14, обід),
заняття(14, 19, вільний час),
заняття(19, 20, вечеря),
заняття(20, 23, відпочинок),
заняття(23, 24, сон).

*Сформулювати запитання:*

Коли буде обід?

Що буде між 14 та 19 годинами?

Коли буде сон?

*Програма має такий вигляд*:

```
implement main
    open core,console
domains %оголошення типів
    sym = symbol.
    int = integer.

class predicates %оголошення правил
    find:(sym) procedure(i).
    find:(int,int)procedure(i,i).

class facts %оголошення фактів
    заняття:(int,int,sym). %факт «заняття» з двома цілочисловими та одним
                           символьним параметром
clauses %пропозиції до факту «заняття»
    заняття(0, 7, "сон").
    заняття(7, 8, "сніданок").
    заняття(8, 13, "школа").
    заняття(13, 14,"обід обід").
    заняття(14, 19, "вільний час").
    заняття(19, 20, "вечеря").
    заняття(20, 23, "відпочинок").
    заняття(23, 24, "сон").

   find(X):- %процедура для пошуку пропозицій «заняття» по третьому параметру

        foreach заняття(A,B,X) do %перебір пропозицій у циклі, де A та B –
                                  вихідні параметри, X – вхідні
        write(X,": ",A,"-",B,"\n") %виведення отриманих значень
        end foreach.

    find(A,B):- %пошук по першим двом параметрам
        foreach заняття(A,B,X) do  %A та B – вхідні
```

```
            write(X,": ",A,"-",B,"\n")
        end foreach.

    run() :-
    init(), %ініціалізація консолі

   find("обід обід"), %виклик процедури
   find(14,19),
    find("сон"),
   _=readLine(),%затримка результату на екрані
        succeed. %place your own code here
end implement main

goal
    console::runUtf8(main::run).
```

*Отримаємо такий результат*:

обід: 13-14
вільний час: 14-19
сон: 0-7
сон: 23-24

Для правильного виведення результату з кирилицею необхідно під час виконання програми встановити властивості Consoles (на вікні чорного кольору при виконанні програми) та розмір шрифту для консольного режиму. Для імен предикатів можна використовувати кирилицю.

Як бачимо, розв'язання завдання виконується шляхом перебору усіх фактів розділу `clauses` за допомогою оператора циклу `foreach`.

**Завдання № 2.**

Знайти площу трапеції з основами A, B і висотою H.

*Описати предикат для обчислення функції, заданої виразом*:

$$f(x)=\begin{cases} x^2, & x<-1, \\ x+1, & -1<x<1, \\ x^2, & x>1. \end{cases}$$

*Листінг програми має вигляд*:

```
implement main
    open core,console

class predicates
    трапеція: (real,real,real,real) procedure(i,i,i,o). %оголошен
ня процедури с трьома вхідними дійсними числами та одним вихідним
    функція:(real,real) procedure(i,o).
clauses
    трапеція(A,B,H,Y):-
        Y = (A+B)/2 * H. %запис результату виразом (A+B)/2*H у вихідну змінну
    функція(X,Y):-
    if X < -1 or X > 1 %перевірка умови X<-1 або X>1
        then  Y = X*X %запис результату X*X у Y
    else %інакше
        if X>-1 and X<1 %перевірка умови
            then Y = X+1
        else Y = 0
        end if
    end if.

    run() :-
        init(),
        трапеція(6,8,2.5,Y), %виклик процедури, яка повертає результат
                                    у змінну Y
        write(Y,"\n"),
        функція(0.2,I), %виклик процедури, яка повертає результат у змінну I
        write( I),
        _=readLine(),%
        succeed.
end implement main
goal
    console::runUtf8(main::run).
```

*Отримаємо такий результат*:

17.5

1.2

Бачимо, що обчислення реалізовано побудовою виразів у вигляді правил Visual Prolog, які оформлено як процедури з вказівкою типу параметрів (вхідні, вихідні, цілі, дійсні і т.д.).

Виклик функцій реалізовано як послідовність з вказівкою фактичних параметрів у правилі `run()`. Виведення результату виконується консольною функцією `write()`.

### 2.5.3 Завдання для індивідуального виконання роботи

Пропонується змоделювати роботу зі своєю власною базою знань та обчислити значення вибраних функцій за логічним виразом та формулою.

### 2.5.4 Контрольні запитання та завдання

1. Пояснити, для чого слугує директива open core, console.
2. Пояснити, Visual Prolog 7.5 – це компілятор чи інтерпретатор.
3. Пояснити, в який спосіб запускається програма у консольному режимі.
4. Пояснити типи параметрів процедури `procedure(i,i,i,o)`.
5. Пояснити, як організувати введення та виведення даних у консольному режимі.
6. Пояснити, чим принципово відрізняється програмування у Visual Prolog від традиційних мов.
7. Проаналізувати способи послідовного аналізу фактів у розділі `clauses`.

### 2.6 Програмування рекурсивних викликів. Робота зі списками

*Рекурсія* – це спосіб організації обчислювального процесу, при якому функція в ході виконання звертається сама до себе [6].

Функція називається *рекурсивною*, якщо під час її виконання можливий повторний її виклик безпосередньо (прямий виклик) або шляхом виклику іншої функції, в якій міститься звертання до неї (непрямий виклик).

#### 2.6.1 Приклади програмування рекурсивних викликів

На прикладі двох завдань розглянемо особливості мови Visual Prolog при організації рекурсії та роботи зі списками.

**Завдання № 1.**

Обчислити суму чисел від 1 до *n*.

*Листінг програми має такий вид*:

```
implement main
open core,console
class predicates
    sum:(integer,integer) procedure(i,o). %опис предикатів як процедур,
                                          і-вхід, о-вихід
clauses
    sum(1,X) :- !,X = 1. %при першому параметрі, що = 1 повернути X = 1
    sum(N,X) :- sum(N-1,Y), X = N+Y.
     % рекурсивне повернення суми поточного N та результату при N-1
        run() :-
             init(),
        write("Введіть ціле число : "),
        S = readLine(), %читання рядка з клавіатури
        N = toTerm(S), %перетворення рядка у число
        sum(N,X), %виклик процедури
        write("Сума цілих значень від 1 до ",N," = ", X),
        _=readLine(),
        succeed.
end implement main
goal
    console::runUtf8(main::run).
```

Результат виконання програми подано на рисунку 2.9.

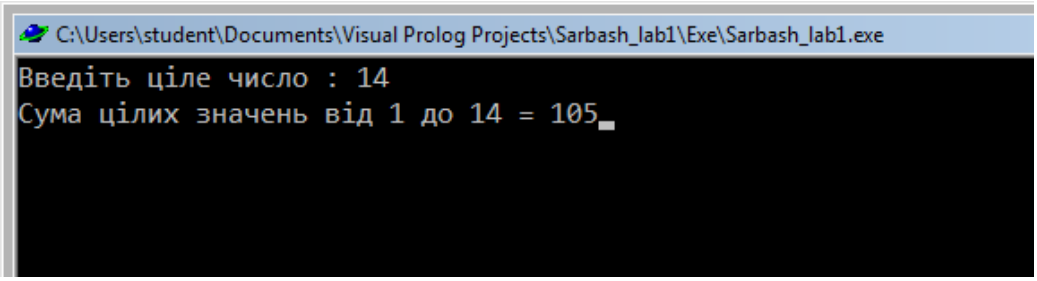



Рисунок 2.9 – Результат виконання програми по завданню № 1

Організовано рекурсивний виклик процедури `sum`, який зупиняється відсіченням `!` при першому параметрі, рівному 1.

**Завдання № 2.**

Визначити, скільки разів заданий елемент входить до списку цілих чисел.

*Листінг програми має такий вид*:

```
implement main
    open core,console
domains
    i = integer.
    list = i*. %оголошення типу списку, який складається із цілих чисел
class predicates
    entries:(list,i,i) procedure(i,i,o).
clauses
    entries([],_,0):-!.  %повернення 0 при отриманні пустого списку
    entries([X|T],Y,K):- %отримання списку з першим елементом X та "хвостом"
                                (того ж списку без першого елемента) T
        entries(T,Y,K1), %перевірка правила для T
        X=Y, %перевірка рівності першого елемента списку та вхідного параметру Y
        K=K1+1,!. %збільшення лічильника на 1
    entries([_|T],Y,K):- %рекурсивна перевірка по T
        entries(T,Y,K).
    run() :-
        init(),
        C=[9,5,9,5,8,4,9,1,56,9,9,8,8],
        write("Вхідний список: ", C, "\n", "Введіть шуканий
елемент списку: "),
        H = readLine(), %читання рядка з клавіатури
        N = toTerm(H),
        entries(C,N,Y),
        write("Кількість входжень числа ",H, " дорівнює ", Y),
_=readLine(),
        succeed.
```

```
end implement main
goal
    console::runUtf8(main::run).
```

Результат виконання програми подано на рисунку 2.10.

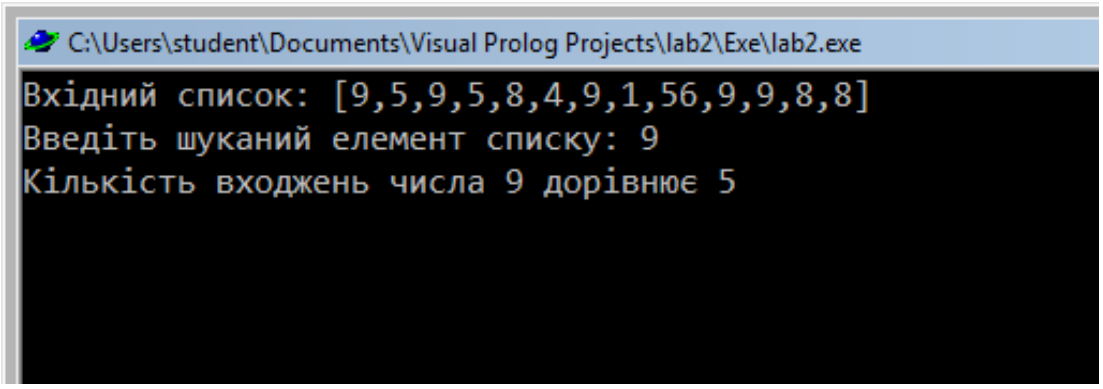



Рисунок 2.10 – Результат виконання програми по завданню № 2

Оброблення списку у Visual Prolog виконується шляхом відділення голови та хвоста списку та продовженням роботи з хвостом, як з новим списком. Відсічення у першому виклику забезпечує зупинку рекурсії, коли список стає пустим. Відсічення у рядку `K=K1+1,!.` забезпечує збільшення лічильника та зупинку у перевірці правила.

### 2.6.2 Завдання для індивідуального виконання роботи

Пропонується обчислити факторіал введеного числа, організувавши рекурсивний виклик. Обчислити суму ряду, де присутній факторіал. Сформувати список елементів та змоделювати роботу зі списком по визначенню загальної кількості елементів, числа додатних елементів та інше.

### 2.6.3 Контрольні запитання та завдання

1. Пояснити, що таке хвостова рекурсія.
2. Дайте визначення рекурсивної процедури.
3. Пояснити, для чого застосовується відсічення.
4. Пояснити, яким чином можна описати список.
5. Пояснити, як виділяється для проведення аналізу окремий елемент списку.
6. Пояснити, що виконує правило: `entries([_|T],Y,K):-entries(T,Y,K).`
7. Пояснити, яким чином вводяться та виводяться елементи списку.

## 2.7 Вирішення логічних задач у Visual Prolog

*Логічне виведення* – остаточна думка про що-небудь, логічний підсумок, зроблений на основі спостережень, міркувань або розгляду певних фактів [5].

### 2.7.1 Приклади вирішення логічних задач

Розглянемо приклади програм, де реалізовано застосування логічних виведень та доведень, для чого, власне, і призначена мова Visual Prolog. При цьому система програмування Visual Prolog робить ці висновки самостійно, спираючись виключно на вказані програмістом факти і правила.

**Завдання № 1.**

Визначити хобі людей через зв’язок з іншими людьми.

*Лістинг програми має вигляд*:

```
implement main
    open core, console
class predicates
    likes: (symbol,symbol) nondeterm.
class predicates
    test: ().
clauses
    likes("ellen" ,"tennis").
    likes("tom", "baseball").
    likes("bill", Activity) :- likes ("tom", Activity) ;
likes("ellen" ,Activity).
    test() :-
        likes("bill", "baseball"),
        !,
        write("Yes");
        write("No").
    run():-
        init(),
        write("Bill like baseball?"),
        test(),
        _ = readChar().
end implement main
goal
    mainExe::run(main::run).
```

Результат виконання програми подано на рисунку 2.11.

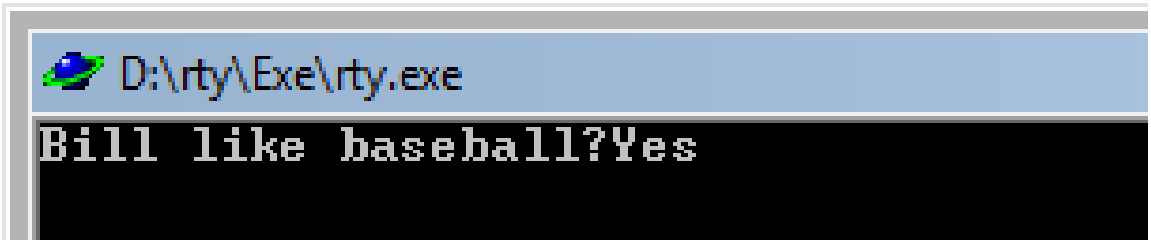



Рисунок 2.11 – Результат виконання програми по завданню № 1

Ключове слово `nondeterm` визначає, що предикат може знаходити декілька рішень. Це дає змогу застосовувати його всередині правил.

`Activity` – це змінна, за допомогою якої організовано перебір наявних фактів та правил задля отримання логічного виведення. Відсічення у процедурі `test` забезпечує завершення виконання програми.

**Завдання № 2.**

Задати предикати: мати, батько, дитина, батьки. Визначити, хто є дитиною та хто є батьками.

*Лістинг програми має вигляд*:

```
implement main
open core, console
class predicates
хтоДитина:() procedure.
хтоБатьки:() procedure.
мати: (string, string) multi(o,o) nondeterm(o,i) nondeterm(i,o).
батько: (string, string) multi(o,o) nondeterm(o,i) nondeterm(i,o).
дитина: (string) nondeterm anyflow.
батьки: (string, string) nondeterm(o,o) nondeterm(i,o).
Clauses
мати("Анна", "Света").
батько("Максим", "Света").
дитина("Володя").
дитина(Дитина) :- мати(Батьки1, Дитина),
батько(Батьки2, Дитина), Батьки1<>Батьки2.
батьки(Батьки1, Батьки2) :- мати(Батьки1, Дитина),
батько(Батьки2, Дитина), Батьки1<>Батьки2.
хтоДитина():-
дитина(X),
```

```
write(X, " - дитина. ", "\n"),
nl,
fail.
хтоДитина().
хтоБатьки():-
батьки(Z, Y),
write(Z, " и ", Y, " - батьки.", "\n"),
nl,
fail.
хтоБатьки().
run() :-
init(),
хтоДитина(),
хтоБатьки(),
_ =readchar().
end implement main
goal
mainExe::run(main::run).
```

Результат виконання програми подано на рисунку 2.12.

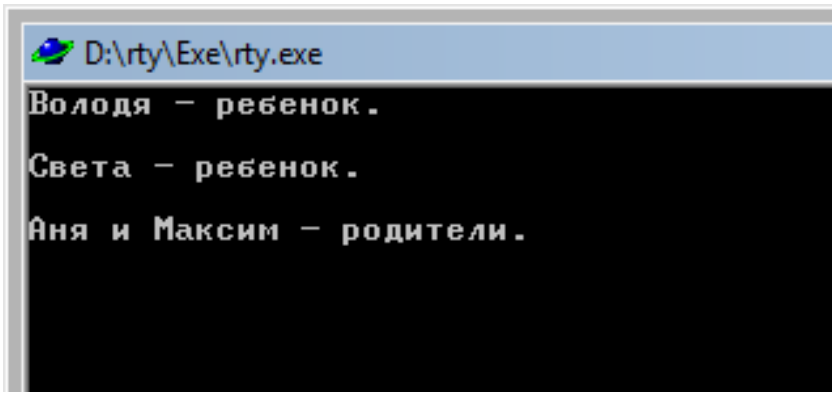



Рисунок 2.12 – Результат виконання програми по завданню № 2

Ключове слово `anyflow` означає, що окремі аргументи предикату можуть бути як вхідними, так і вихідними, компілятор виконує вибір типу автоматично.

Предикат `fail` заставляє програму продовжувати шукати рішення доки не закінчиться вся наявна інформація.

Ключове слово `multi` вказує, що зазначені у дужках параметри у будь-якому випадку будуть знаходитися у вказаному відношенні (наприклад, задаючи відношення `parent("Vasya", "Petya")` маємо на увазі, що Василь син Петра).

### 2.7.2 Завдання для індивідуального виконання роботи

1. Задайте у програмі родинні бінарні відношення: внук, внучка, дідусь, бабуся, син, дочка, сестра, брат. Визначте відношення сестра, брат через інші.

2. Напишіть програму, яка за допомогою бінарних відношень: володар, тварина, колір та 5-6 відповідних фактів реалізує запити: знайти володарів сірих котів, знайти тварину Ірини та її колір, знайти тварин, якими володіють Володимир та Олексій, знайти володарів тварин оранжевого кольору.

### 2.7.3 Контрольні запитання та завдання

1. Пояснити суть логічного виведення.
2. Пояснити, для чого використовується опис `nondeterm`.
3. Пояснити призначення предикату `fail`.
4. Розшифрувати правило виведення:

птах (Об’єкт): – має_крила (Об’єкт), вміє_літати (Об’єкт); каркає(Об’єкт).

5. Пояснити, якому вислову в природній мові відповідає факт Visual Prolog з використанням анонімної змінної: носить ( ¬_, взуття).
6. Пояснити дію фрагмента програми на Visual Prolog:

```
class predicates
    find:(sym) procedure(i).
    find:(int,int)procedure(i,i).
```

7. Пояснити, яку дію виконує фрагмент програми:

```
предок (A, B) : -
            батько (A, B).
предок (A, B) : -
            батько(C, B),
            предок (A, C).
```

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ


1. Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. Методы и модели анализа данных: OLAP и Data Mining: учеб. пособие. СПб: БХВ-Петербург. 2004. 336 с.

2. Ланде Д.В., Субач І.Ю., Бояринова Ю.Є. Основи теорії і практики інтелектуального аналізу даних у сфері кібербезпеки: навч. посібник. Київ: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2018. 297 с.

3. Гладун Ф.Я., Рогушина Ю.В. Data Mining: пошук знань в даних: підручник. Київ: ТОВ «ВД «АДЕФ-Україна», 2016. 452 с.

4. Вуколов В.А. Основы статистического анализа. Практикум по статистическим методам и исследованию операций с использованием пакетов STATISTICA и EXEL: учеб. Пособие. М.: ФОРУМ, 2008. 464 с.

5. Нильс Нильсон. Искусственный интеллект. Методы поиска решений. М. Мир, 1973. 270 с.

6. Бондарев В.Н., Аде Ф.Г. Искусственный интеллект: учеб. пособие. Севастополь: СевНТУ, 2002. 615 с.

7. Бодянский Е.В., Кучеренко Е.И., Михалев А.И., Гасик М.М., Куцин В.С., Филатов В.А. Интеллектуальное управление технологическими процессами: монография. Днепропетровск: Национальная металлургическая академия Украины, 2013. 213 с.

8. Макаров И.М., Виноградская Т.М., Рубчинский А.А., Соколов В.Б. Теория выбора и принятия решений: учеб. пособие. М.: Наука, 1982. 328 с.

9. Айвазян С.А., Бухштабер В.М., Енюков И.С., Мешалкин Л.Д. Прикладная статистика: Классификация и снижение размерности: справочное издание / под. ред. С.А. Айвазян. М.: Финансы и статистика, 1989. 607 с.

10. Алгоритмы кластеризации на службе Data Mining. URL: http://www.olap.ru/home.asp?artId=154.

11. Путятін Є.П., Гороховатський В.О., Матат О.О. Методи та алгоритми комп'ютерного зору: навч. посібник. Харків: Компанія СМІТ, 2006. 236 с.

12. Мацуга О.М., Архангельська Ю.М., Єрещенко Н.М. Навчальний посібник до вивчення курсу «Інформаційні технології розпізнавання образів». Дніпропетровськ: РВВ ДНУ, 2016. 60 с.

13. Путятин Е.П., Аверин С.И. Обработка изображений в робототехнике. М.: Машиностроение, 1990. 320 с.

14. Шапиро Л., Стокман Дж. Компьютерное зрение: учеб. пособие. М.: Бином, 2006. 752 с.

15. Шумейко А.А., Сотник С.Л. Интеллектуальный анализ данных (Введение в Data Mining): учеб. пособие. Днепропетровск: Белая Е.А., 2012. 212 с.

16. Довбиш А.С., Шелехов І.В. Основи теорії розпізнавання образів: навч. посібник. Суми: Сумський державний університет, 2015. Ч. 1. 109 с.

17. Daradkeh Y.I., Tvoroshenko I., Gorokhovatskyi V., Latiff L.A., Ahmad, N. Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic. *IEEE Access*. 2021. vol. 9. pp. 13417-13428.

18. Claus C., Huitl R., Rausch J., Stechele W. Optimizing the SUSAN corner detection algorithm for a high speed FPGA implementation. *2009 International Conference on Field Programmable Logic and Applications*. Prague. 2009. pp. 138-145.

19. Кондратенко Ю.П., Мухортова К.В. Цикл практичних робіт із дисципліни «Теорія нечітких множин. Нечітка логіка». Миколаїв: МДГУ імені П. Могили, 2005. 39 с.

20. Olena Vynokurova, Dmytro Peleshko, Marta Peleshko Hybrid Deep Convolutional Neural Network with Multimodal Fusion. In: Babichev S., Peleshko D., Vynokurova O. (eds) *Data Stream Mining & Processing. DSMP 2020*. Communications in Computer and Information Science. Springer, Cham. 2020. vol. 1158. pp. 62-78.

21. Visual Prolog 7.5 для початківців. URL: http://primat.org/publ/prolog/visual_prolog_7_5_dlja_novichkov_urok_3_osnovnye_razdely_programmy/65-1-0-1436

22. Офіційний сайт Visual Prolog. URL: http://www.visual-prolog.com/vip/download/

23. Братко И. Алгоритмы искусственного интеллекта на языке PROLOG. М.: Издательский дом «Вильямс», 2004. 640 с.